Warpgate: Bastion host moderno e gestione degli accessi con SSO Authentik

Warpgate: Bastion host moderno e gestione degli accessi con SSO Authentik

La gestione distribuita degli accessi amministrativi SSH, database e console web su infrastrutture multi-server rappresenta una delle principali sfide della sicurezza perimetrale. L'assenza di un controllo centralizzato delle credenziali favorisce la proliferazione di chiavi statiche e rende complessa la revoca puntuale delle autorizzazioni.

Per implementare una governance degli accessi robusta, scalabile e completamente open source, **Warpgate** rappresenta una soluzione architetturale di riferimento.

Warpgate è un bastion host moderno scritto in Rust che opera come proxy trasparente per protocolli SSH, HTTP, MySQL, Postgres e cluster Kubernetes. Tra le funzionalità salienti figura il supporto nativo al Single Sign-On (SSO) mediante OpenID Connect (OIDC). Di seguito viene analizzata la procedura di installazione tramite Docker, il reverse proxying tramite Nginx e l'integrazione del controllo accessi basato su ruoli (RBAC) con Authentik.

Confronto architetturale e vantaggi operativi

Prima di sporcarci le mani con la configurazione, è bene capire dove si colloca questo strumento rispetto alle altre soluzioni di accesso sul mercato:

- **SSH Jump Host (ProxyJump)**: La soluzione minimale integrata in OpenSSH. È gratuita ed estremamente semplice, ma è limitata al solo SSH, non offre un'autenticazione SSO integrata (a meno di non

voler impazzire con i moduli PAM/SSSD) e non registra le sessioni in modo nativo.
- **Teleport**: Un'ottima piattaforma d'accesso in Go, ma decisamente sovra-ingegnerizzata per piccoli e medi team. La versione Community è castrata su diverse funzioni enterprise e costringe gli utenti a usare il loro client proprietario (`tsh`) anziché i normali comandi nativi, appesantendo il flusso di lavoro quotidiano.
- **CyberArk (PAM)**: Il dinosauro proprietario del mondo corporate. Estremamente costoso, richiede un'infrastruttura titanica basata su server Windows dedicati e database, ed è una piaga burocratica per chiunque debba interfacciarsi quotidianamente con le macchine.
- **Warpgate**: Si colloca nel mezzo. È 100% open-source, non richiede client dedicati (usi i tuoi soliti software), integra l'SSO OIDC ed è leggerissimo da far girare in un semplice container.

Bando alle ciance e iniziamo.

Passo 1: L'installazione (il buon vecchio Docker)

Creiamo la directory di progetto sul server:

```
# mkdir -p /opt/warpgate && cd /opt/warpgate
```

Prepariamo il nostro `docker-compose.yml`:

```
services:
  warpgate:
    image: ghcr.io/warp-tech/warpgate:latest
    container_name: warpgate
    restart: unless-stopped
    ports:
      - "2222:2222"      # SSH in ingresso
      - "8888:8888"      # Admin Web UI / HTTPS proxy
      - "33306:33306"    # MySQL proxy
      - "55432:55432"    # PostgreSQL proxy
    volumes:
      - ./data:/data     # Configurazione, database sqlite e registrazioni sessioni
```

```
    stdin_open: true
    tty: true
```

Prima di avviare il container, dobbiamo lanciare la configurazione guidata iniziale per generare le chiavi SSH del bastion e il database SQLite interno:

```
# docker compose run --rm warpgate setup
```

Segui i passaggi a schermo impostando una password provvisoria per l'amministratore locale. Questo comando creerà la cartella `./data` con il file `warpgate.yaml`.

Passo 2: Mettere Warpgate dietro Nginx (e scontrarsi con il TLS)

Se vuoi esporre l'interfaccia web di Warpgate su `warpgate.example.org`, devi configurare un reverse proxy. Ma ecco la prima stranezza: all'avvio, Warpgate genera dei certificati auto-firmati e attiva il TLS sulla porta web `8888`.

Se provi a configurare Nginx con un classico `proxy_pass http://127.0.0.1:8888;`, riceverai un bel `upstream sent no valid HTTP/1.0 header`. Nginx sta parlando in chiaro a una porta che si aspetta la cifratura.

La cura è forzare Nginx a parlare HTTPS verso il backend, disabilitando la verifica del certificato interno auto-firmato:

```
server {
    listen 80;
    server_name warpgate.example.org;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    server_name warpgate.example.org;

    ssl_certificate /etc/letsencrypt/live/warpgate.example.org/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/warpgate.example.org/
privkey.pem;
```

```
    client_max_body_size 4G;

    location / {
        # HTTPS obbligatorio e porta 8888
        proxy_pass https://127.0.0.1:8888;
        proxy_ssl_verify off; # Ignora il certificato auto-generato interno

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Supporto WebSockets per le console interattive
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        proxy_connect_timeout 600s;
        proxy_read_timeout 600s;
        proxy_send_timeout 600s;
    }
}
```

Esegui un `nginx -t` e ricarica il servizio.

Passo 3: Configurazione su Authentik (OIDC)

Spostiamoci sull'istanza di Authentik (`auth.example.org`) per configurare l'applicazione.

1. **Crea il Provider OIDC**:
    - Vai su **Applications** > **Providers** > **Create**.
    - Seleziona **OAuth2/OpenID Provider**.
    - Nome: `warpgate`.
    - Client Type: `Confidential`.
    - Redirect URI: `https://warpgate.example.org/@warpgate/api/sso/return` (senza indicare la porta `8888` se Nginx risponde sulla 443).
    - Prendi nota di **Client ID** e **Client Secret**.
2. **Crea l'Applicazione**:
    - Vai su **Applications** > **Applications** > **Create**.

    - Nome: `Warpgate`, Slug: `warpgate`.
    - Seleziona il Provider creato al punto precedente.

I trucchi OIDC (Property Mappings obbligatori)

Se provi ad accedere adesso, ti scontrerai con due blocchi di sicurezza di Warpgate. Risolviamoli subito creando dei **Property Mapping** di tipo **Scope** in Authentik (**Customization** > **Property Mappings**):

1. Forzare l'email verificata (`warpgate-email-verified`)

Nelle ultime versioni, Authentik invia il claim `email_verified` come `false` di default. Warpgate rifiuterà il login. Crea una mappatura con:

- **Scope name**: `email`
- **Expression**:

```
return {
    "email": request.user.email,
    "email_verified": True,
}
```

2. Mappare i ruoli amministrativi (`warpgate-admin-roles-mapping`)

Warpgate distingue l'accesso ai target dai privilegi amministrativi sul portale. Cerca il claim `warpgate_admin_roles` nel token. Creiamo una mappatura per dire ad Authentik che gli utenti del gruppo `WARP-Admins` sono amministratori di Warpgate:

- **Scope name**: `openid`
- **Expression**:

```
admin_roles = []
for group in user.groups.all():
    if group.name == "WARP-Admins":
        admin_roles.append("warpgate:admin")
return {
    "warpgate_admin_roles": admin_roles
}
```

3. Mappare i ruoli di accesso normali (`warpgate-roles-mapping`)

Per dire invece a quali macchine gli utenti possono collegarsi, usiamo il claim `warpgate_roles`:

- **Scope name**: `openid`
- **Expression**:

```
roles = []
for group in user.groups.all():
    if group.name != "WARP-Admins":
        roles.append(group.name) # Passa tutti gli altri gruppi (es. WARP-Netadmins)
return {
    "warpgate_roles": roles
}
```

Assicurati di andare nelle impostazioni del Provider su Authentik e selezionare queste tre mappature nel campo **Property Mappings**.

Passo 4: Configurare `warpgate.yaml`

Ora modifichiamo il file `data/warpgate.yaml` sul server per abilitare l'SSO e dire a Warpgate di fidarsi di Nginx:

```
external_host: warpgate.example.org

sso_providers:
  - name: authentik
    label: "Accedi con Authentik"
    provider:
      type: custom
      client_id: <IL_TUO_CLIENT_ID_AUTHENTIK>
      client_secret: <LA_TUA_CLIENT_SECRET_AUTHENTIK>
      issuer_url: https://auth.example.org/application/o/warpgate/
      scopes:
        - openid
        - email
        - profile
    auto_create_users: true

http:
  listen: '[::]:8888'
```

```
  external_port: 443
  external_host: warpgate.example.org
  certificate: /data/tls.certificate.pem
  key: /data/tls.key.pem
  trust_x_forwarded_headers: true # Fondamentale per far leggere gli header a
Nginx
  session_max_age: 30m
  cookie_max_age: 1day
  sni_certificates: []
```

Salva il file e riavvia il container:

```
# docker compose restart warpgate
```

Come collegarsi alle macchine target

Ora che l'SSO funziona e sei loggato come amministratore, puoi configurare i tuoi target (es. un server SSH su porta 22) direttamente dalla Web UI di Warpgate.

Quando configuri un target, inserisci nel campo **Allowed Roles** il nome del gruppo di Authentik che desideri abilitare (es. `WARP-Netadmins`).

Ricorda infine che Warpgate non inoltra la tua chiave SSH privata alla destinazione. Dovrai copiare la chiave pubblica di Warpgate (la trovi nella sezione **SSH** della Web UI o lanciando `docker compose exec warpgate warpgate client-keys`) e appenderla nel file `~/.ssh/authorized_keys` dell'utente sulla macchina remota.

Zero chiavi personali sparse in giro, zero password scritte in chiaro. Tutto gestito centralmente.

[Che il talpone sia con voi]

[EOF]

Fluxer Icon

Canale di discussione e supporto

Canale di discussione e supporto. Per confrontarti su configurazioni o altro entra su fluxer.dreadful.work (https://fluxer.dreadful.work/invite/dreadful).