OWASP Top 10 Pentest-Checkliste

Übersicht

Diese Checkliste basiert auf den OWASP Top 10:2021 und enthält Nago-spezifische Testszenarien für Penetrationstests.

Legende

Symbol	Bedeutung
□	Schutzmaßnahme implementiert
□□	Teilweise implementiert oder bekanntes Risiko
□	Nicht implementiert
N/A	Nicht anwendbar

A01:2021 - Broken Access Control

Testszenarien

ID	Testszenario	Status	Hinweise
A01-01	Permission Bypass via Session-Cookie-Manipulation	□	`subject.Audit()` prüft bei jedem Aufruf
A01-02	Horizontale Privilege Escalation (Zugriff auf fremde Ressourcen)	□	`subject.AuditResource()` für ressourcenspezifische Prüfung
A01-03	Vertikale Privilege Escalation (Admin-Funktionen ohne Berechtigung)	□	Permission-System mit Audit-Trail
A01-04	Repository-ID-Escalation mit `nago.*` Präfix	□	Security Note: Validierung in `flow/evt_repository_assigned.go:27`
A01-05	User Circle Bypass	□	Security Note: UI-Schutz in `usercircle/ui/view_users.go:32`
A01-06	CORS-Bypass	□□	Debug-Modus hat offene CORS-Policy

Spezifische Tests

Test A01-01: Permission Bypass

```
1. Authentifizieren Sie sich als normaler Benutzer
```

```
2. Notieren Sie die Session-ID aus dem Cookie
3. Versuchen Sie, Admin-Endpoints aufzurufen:
   - /admin/iam/user
   - /admin/iam/role
   - /admin/backup
4. Erwartetes Ergebnis: 403 Forbidden oder leere Antwort
```

Test A01-04: Repository-ID-Escalation

```
1. Authentifizieren Sie sich mit Flow-Berechtigung
2. Versuchen Sie, ein Repository mit ID "nago.iam.user" zu erstellen
3. Erwartetes Ergebnis: Validation Error
4. Versuchen Sie ID "nago.iam.role", "nago.session"
5. Alle sollten abgelehnt werden
```

A02:2021 - Cryptographic Failures

Testszenarien

ID	Testszenario	Status	Hinweise
A02-01	Master Key Exposure via Environment	□	Prüfen Sie `/proc/[pid]/environ` (Linux)
A02-02	Master Key File Permissions	□	`.masterkey` sollte 0600 sein
A02-03	Schwache Passwort-Hashing-Algorithmen	□	Nur Argon2id unterstützt
A02-04	TLS-Version für SMTP	□	`InsecureSkipVerify: false`
A02-05	Backup ohne Master Key	□□	Verschlüsselte Stores bleiben verschlüsselt

Spezifische Tests

Test A02-01: Master Key Exposure

```
# Auf dem Server ausführen:
cat /proc/$(pgrep nago)/environ | tr '\0' '\n' | grep NAGO_MASTER_KEY

# Erwartetes Ergebnis:
# - Entweder nicht gesetzt (Datei-Fallback)
# - Oder nur für root/Prozess-Owner sichtbar
```

Test A02-02: File Permissions

```
ls -la /path/to/data/.masterkey
```

```
# Erwartetes Ergebnis: -rw------- (0600)
```

A03:2021 - Injection

Testszenarien

ID	Testszenario	Status	Hinweise
A03-01	SQL Injection	N/A	Blob Storage, kein SQL
A03-02	XSS in VueJS-Komponenten		Vue escapet standardmäßig
A03-03	Command Injection in File-Uploads		MIME-Type-Validierung, Magic-Byte-Prüfung
A03-04	Staging-ID-Injection		Security Note: `dataimport/page_select_parser.go:72`
A03-05	Log Injection		Strukturiertes Logging mit slog

Spezifische Tests

Test A03-02: XSS-Test

```
1. Versuchen Sie folgenden Payload in Textfeldern:
   <script>alert('XSS')</script>

2. Versuchen Sie in Benutzernamen/Profilfeldern:
   <img src=x onerror=alert('XSS')>

3. Erwartetes Ergebnis: HTML wird escaped angezeigt
```

Test A03-04: Staging-ID-Injection

```
1. Navigieren Sie zum Data Import
2. Versuchen Sie, Query-Parameter hinzuzufügen:
   /admin/dataimport?stagingId=existing-staging-id

3. Erwartetes Ergebnis: Parameter wird ignoriert
```

A04:2021 - Insecure Design

Testszenarien

ID	Testszenario	Status	Hinweise
A04-01	Session Fixation	☐	Neue Session-ID nach Authentifizierung
A04-02	CSRF via WebSocket	☐☐	SameSite-Cookie, aber WebSocket-Prüfung TODO
A04-03	Race Conditions in Permission-Checks	☐	Mutex-Lock für kritische Operationen
A04-04	WebSocket Origin-Validierung	☐☐	`CheckOrigin: true` - TODO im Code

Spezifische Tests

Test A04-02: WebSocket CSRF

```
1. Erstellen Sie eine externe HTML-Seite:
   <script>
     var ws = new WebSocket('wss://target.com/wire?_sid=attacker-scope');
     ws.onopen = function() {
       ws.send('{"type":"FunctionCallRequested",...}');
     };
   </script>

2. Öffnen Sie die Seite während Sie bei target.com angemeldet sind
3. Prüfen Sie, ob die Anfrage akzeptiert wird
4. Hinweis: CheckOrigin ist aktuell deaktiviert!
```

A05:2021 - Security Misconfiguration

Testszenarien

ID	Testszenario	Status	Hinweise
A05-01	Debug-Mode in Production	☐☐	CORS-Einstellungen prüfen
A05-02	Default Bootstrap-Admin	☐	Zeitlimit, Audit-Log
A05-03	Exposed Error Messages	☐	Lokalisierte Fehlermeldungen
A05-04	Directory Listing	☐	Kein automatisches Listing

Spezifische Tests

Test A05-01: Debug-Mode-Check

```
1. Senden Sie eine OPTIONS-Anfrage:
   curl -X OPTIONS https://target.com/api -v
```

```
2. Prüfen Sie CORS-Header:
   - Access-Control-Allow-Origin: *  <- KRITISCH!

3. Wenn Allow-Origin "http://*" enthält, ist Debug-Mode aktiv
```

Test A05-02: Bootstrap-Admin

```
1. Versuchen Sie Login mit:
   - Email: admin@localhost
   - Password: (Default aus Dokumentation)

2. Erwartetes Ergebnis:
   - Entweder abgelehnt (Zeitlimit abgelaufen)
   - Oder funktioniert (KRITISCH - muss deaktiviert werden)
```

A06:2021 - Vulnerable and Outdated Components

Testszenarien

ID	Testszenario	Status	Hinweise
A06-01	Go Dependency Vulnerabilities		`govulncheck ./⋯`
A06-02	NPM/VueJS Vulnerabilities		`npm audit` in web/vuejs
A06-03	Veraltete Krypto-Libraries		golang.org/x/crypto aktuell

Spezifische Tests

Test A06-01: Go Vulnerability Scan

```
cd /path/to/nago
go install golang.org/x/vuln/cmd/govulncheck@latest
govulncheck ./...
```

Test A06-02: NPM Audit

```
cd /path/to/nago/web/vuejs
npm audit
npm audit --production
```

A07:2021 - Identification and Authentication Failures

Testszenarien

ID	Testszenario	Status	Hinweise
A07-01	Brute-Force-Angriff	□□	Kein globales Rate-Limiting
A07-02	Credential Stuffing	□□	Kein Captcha, aber Argon2id macht es teuer
A07-03	Password Reset Token Reuse	□	Security Note: Neuer Code bei jedem Reset
A07-04	Session Timeout	□	3 Monate, konfigurierbar
A07-05	E-Mail-Verifikation Bypass	□	Pflicht-Verifikation vor Login
A07-06	User Enumeration bei Registrierung	□□	Akzeptiertes Risiko, dokumentiert

Spezifische Tests

Test A07-01: Brute-Force

```
1. Verwenden Sie ein Tool wie Hydra:
   hydra -l user@example.com -P wordlist.txt target.com https-post-form \
     "/api/login:email=^USER^&password=^PASS^:F=incorrect"

2. Beobachten Sie:
   - Werden Anfragen verzögert?
   - Wird das Konto gesperrt?

3. Hinweis: Argon2id macht jeden Versuch teuer (19 MiB Memory)
```

Test A07-03: Password Reset Token Reuse

```
1. Fordern Sie einen Password-Reset an
2. Notieren Sie den Link/Token
3. Setzen Sie das Passwort zurück
4. Versuchen Sie, den gleichen Token erneut zu verwenden
5. Erwartetes Ergebnis: Token ist ungültig
```

A08:2021 - Software and Data Integrity Failures

Testszenarien

ID	Testszenario	Status	Hinweise
A08-01	Backup-Manipulation	□□	Keine kryptographische Signatur
A08-02	Event-Log-Tampering	□	Append-Only Event Store
A08-03	Unsigned Software Updates	□□	Keine automatischen Updates

Spezifische Tests

Test A08-01: Backup-Integrity

```
1. Erstellen Sie ein Backup
2. Entpacken Sie die ZIP-Datei
3. Modifizieren Sie eine JSON-Datei
4. Packen Sie die ZIP-Datei neu
5. Führen Sie einen Restore durch
6. Prüfen Sie, ob die Manipulation erkannt wird
7. Erwartetes Ergebnis: □□ Aktuell keine Signatur-Prüfung
```

A09:2021 - Security Logging and Monitoring Failures

Testszenarien

ID	Testszenario	Status	Hinweise
A09-01	Log Injection	□	Strukturiertes Logging mit slog
A09-02	Sensitive Data in Logs	□□	Manuelle Prüfung erforderlich
A09-03	Fehlende Login-Failure-Logs	□	Events werden geloggt
A09-04	Log-Rotation	N/A	Deployment-Ebene

Spezifische Tests

Test A09-02: Sensitive Data in Logs

```
1. Führen Sie verschiedene Aktionen durch:
   - Login (erfolgreich/fehlgeschlagen)
```

```
   - Password-Reset
   - Benutzer-Erstellung

2. Prüfen Sie die Logs auf:
   - Passwörter im Klartext
   - Session-Tokens
   - Master Key

3. Erwartetes Ergebnis: Keine sensitiven Daten geloggt
```

A10:2021 - Server-Side Request Forgery (SSRF)

Testszenarien

ID	Testszenario	Status	Hinweise
A10-01	WebSocket Origin SSRF	□□	CheckOrigin deaktiviert (TODO)
A10-02	File Import SSRF	□	Lokale Datei-Validierung
A10-03	Drive User-ID Access	□□	Security Note: `drive/uc_read_drives.go:38`

Spezifische Tests

Test A10-01: WebSocket Origin

```
1. Erstellen Sie eine WebSocket-Verbindung von einer fremden Origin:
   // Von attacker.com:
   new WebSocket('wss://target.com/wire?_sid=test')

2. Prüfen Sie, ob die Verbindung akzeptiert wird
3. Hinweis: CheckOrigin ist aktuell deaktiviert!
```

Zusammenfassung

Kritische Findings

Kategorie	Finding	Empfehlung
A04	WebSocket Origin-Validierung deaktiviert	CheckOrigin-Funktion implementieren
A05	Debug-Mode CORS in Production möglich	Automatische Deaktivierung in Production

Kategorie	Finding	Empfehlung
A07	Kein globales Rate-Limiting für Login	Rate-Limiter implementieren

Akzeptierte Risiken

Kategorie	Risiko	Begründung
A07-06	User Enumeration bei Registrierung	UX-Trade-off, dokumentiert
A08-01	Backup ohne Signatur	Master Key bietet teilweisen Schutz

Pentest-Empfehlungen

1. **Fokus auf WebSocket-Sicherheit**: Die WebSocket Origin-Validierung ist ein kritischer Punkt
2. **CORS-Konfiguration prüfen**: Sicherstellen, dass Debug-Mode nicht in Production aktiv ist
3. **Rate-Limiting testen**: Brute-Force-Resistenz analysieren
4. **Backup-Integrität**: Manipulation von Backup-Dateien testen