Nago Low-Code Framework - Sicherheitsdokumentation

Table of Contents

Diese Dokumentation beschreibt die Sicherheitsarchitektur, implementierte Kontrollen und Compliance-Nachweise des Nago Low-Code Frameworks für Sicherheitsaudits nach ISO 27001 und Penetrationstests.

1. Executive Summary

Das **Nago Low-Code Framework** ist eine Enterprise-Grade-Plattform zur schnellen Entwicklung von Geschäftsanwendungen, entwickelt von der worldiety GmbH. Das Framework basiert auf einer modernen Architektur mit Go-Backend und VueJS-Frontend, verbunden über WebSocket-Kommunikation.

1.1. Sicherheits-Highlights

Bereich	Implementierung
Authentifizierung	Argon2id-Passwort-Hashing (OWASP-konform), Session-basierte Authentifizierung, SSO via OAuth2/OpenID Connect
Autorisierung	Role-Based Access Control (RBAC), Permission-basierte Zugriffskontrolle mit Audit-Trail
Kryptographie	AES-GCM-256 für Datenverschlüsselung, 32-Byte Master Key, TLS für Netzwerkkommunikation
Datenschutz	DSGVO-konforme Consent-Verwaltung, E-Mail-Verifikationspflicht, Backup-Verschlüsselung
Logging	Strukturiertes Logging mit slog, Event-basierter Audit-Trail, Session-Tracking

1.2. Dokumentstruktur

Diese Dokumentation ist modular aufgebaut:

- Sicherheitsarchitektur - Systemübersicht, Komponenten, Datenflüsse
- ISO 27001 Controls - Annex A Mapping mit Code-Referenzen
- Bedrohungsmodell - STRIDE-Analyse mit Mitigationen
- Pentest-Checkliste - OWASP Top 10 Testszenarien
- Compliance-Nachweise - ISO 27001, SOC 2, DSGVO Evidences
- Incident Response Plan - Reaktionsverfahren für Sicherheitsvorfälle

2. Geltungsbereich

2.1. In Scope

- Nago Framework Application Layer (Go Backend, VueJS Frontend)
- Session Management und Authentifizierung
- Permission- und Role-Management
- Kryptographische Implementierungen
- Backup- und Restore-Funktionalität
- Event-basiertes Audit-Logging

2.2. Out of Scope

- Deployment-Infrastruktur (siehe Nago Hub Hosting Dokumentation)
- Reverse Proxy / Load Balancer Konfiguration
- TLS-Zertifikatsverwaltung
- Firewall-Regeln
- Container-Orchestrierung
- Host-Betriebssystem-Sicherheit

Für Deployment-Security-Aspekte verweisen wir auf die separate **Nago Hub Hosting Dokumentation**, die alle infrastrukturbezogenen Sicherheitsmaßnahmen beschreibt.

3. Technologie-Stack

Komponente	Technologie	Version
Backend	Go	1.25.0
Frontend	VueJS	3.x
HTTP Router	chi/v5	5.2.3
WebSocket	gorilla/websocket	1.5.3
Kryptographie	golang.org/x/crypto	0.45.0
Passwort-Validierung	wagslane/go-password-validator	0.3.0
Internationalisierung	worldiety/i18n	aktuell

4. Versionierung

Version	Datum	Änderungen	Autor
1.0	2026-02-02	Initiale Version für ISO 27001 Audit	worldiety GmbH

5. Kontakt

Anfragen	Kontakt
Security Issues	security@worldiety.de
Compliance-Fragen	compliance@worldiety.de
Technische Dokumentation	https://nago.dev

6. Glossar

Begriff	Definition
Argon2id	Memory-hard Passwort-Hashing-Algorithmus, empfohlen von OWASP für Passwort-Speicherung
AES-GCM-256	Advanced Encryption Standard mit Galois/Counter Mode und 256-Bit Schlüssel
RBAC	Role-Based Access Control - Rollenbasierte Zugriffskontrolle
Master Key	32-Byte kryptographischer Schlüssel zur Verschlüsselung aller sensitiven Daten
NLS	Nago Login Service - SSO-Provider für OAuth2/OpenID Connect
Event Bus	Internes Publish-Subscribe-System für lose gekoppelte Kommunikation
Blob Storage	Key-Value-basiertes Speichersystem für strukturierte und unstrukturierte Daten
Subject	Authentifizierter Benutzerkontext mit Berechtigungen und Audit-Fähigkeiten

7. Sicherheitsarchitektur

7.1. Systemübersicht

Das Nago Framework implementiert eine mehrschichtige Sicherheitsarchitektur mit klarer Trennung von Verantwortlichkeiten.

7.1.1. C4 System Context Diagramm

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-
PlantUML/master/C4_Context.puml

title System Context - Nago Low-Code Framework

Person(user, "Benutzer", "Anwendungsbenutzer mit Browser")
```

```
Person(admin, "Administrator", "System-Administrator")

System(nago, "Nago Application", "Low-Code Framework\nGo Backend + VueJS Frontend")

System_Ext(nls, "Nago Login Service", "OAuth2/OIDC Provider\nSingle Sign-On")
System_Ext(smtp, "SMTP Server", "E-Mail-Versand\nVerifikation, Benachrichtigungen")
System_Ext(ai, "AI Provider", "Mistral AI, OpenAI\nOptionale KI-Integration")

Rel(user, nago, "Verwendet", "HTTPS/WebSocket")
Rel(admin, nago, "Administriert", "HTTPS/WebSocket")
Rel(nago, nls, "SSO Authentication", "HTTPS/OAuth2")
Rel(nago, smtp, "Sendet E-Mails", "TLS/SMTP")
Rel(nago, ai, "KI-Anfragen", "HTTPS/REST")

@enduml
```

7.1.2. C4 Container Diagramm

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-
PlantUML/master/C4_Container.puml

title Container Diagramm - Nago Application

Person(user, "Benutzer", "Browser")

System_Boundary(nago, "Nago Application") {
    Container(spa, "VueJS SPA", "TypeScript/Vue 3", "Single Page Application\nUI-
Rendering, State Management")
    Container(backend, "Go Backend", "Go 1.25", "Business Logic\nSession
Management\nPermission System")
    Container(ws, "WebSocket Handler", "gorilla/websocket", "Bidirektionale
Kommunikation\n/wire Endpoint")
    ContainerDb(blob, "Blob Storage", "File System", "Verschlüsselte
Datenspeicherung\nAES-GCM-256")
    Container(eventbus, "Event Bus", "Go Channels", "Publish-Subscribe\nAudit Events")
}

Rel(user, spa, "Lädt", "HTTPS")
Rel(spa, ws, "WebSocket", "wss://")
Rel(ws, backend, "Events", "Internal")
Rel(backend, blob, "Read/Write", "Encrypted")
Rel(backend, eventbus, "Publish", "Async")

@enduml
```

7.1.3. Komponenten-Diagramm

```
@startuml
!include https://raw.githubusercontent.com/plantuml-stdlib/C4-
PlantUML/master/C4_Component.puml

title Sicherheitskomponenten - Nago Backend

Container_Boundary(backend, "Go Backend") {
    Component(session, "Session Management", "application/session", "Cookie-basierte
Sessions\n3 Monate TTL\nHttpOnly/Secure")
    Component(user, "User Management", "application/user",
"Benutzerregistrierung\nArgon2id Hashing\nE-Mail-Verifikation")
    Component(permission, "Permission System", "application/permission", "Deklarative
Permissions\nAudit-Trail\nRBAC")
    Component(role, "Role Management", "application/role", "Rollenbasierte
Zugriffskontrolle\nPermission-Aggregation")
    Component(crypto, "Crypto Layer", "pkg/blob/crypto", "AES-GCM-256\nMaster Key
Management")
    Component(backup, "Backup System", "application/backup", "Verschlüsselte
Backups\nMaster Key Export/Import")
    Component(evs, "Event Store", "application/evs", "Audit Trail\nImmutable Events")
}

Rel(session, user, "Authentifiziert")
Rel(user, permission, "Prüft Berechtigungen")
Rel(permission, role, "Aggregiert aus Rollen")
Rel(user, crypto, "Passwort-Hashing")
Rel(backup, crypto, "Verschlüsselung")
Rel(permission, evs, "Audit Events")

@enduml
```

7.2. Authentifizierung

7.2.1. Session Management

Das Session Management basiert auf Cookie-basierter Authentifizierung mit folgenden Eigenschaften:

Eigenschaft	Wert	Sicherheitsrelevanz
Cookie-Name	`wdy-ora-access`	Nicht erratbar, proprietär
TTL	3 Monate nach letzter Authentifizierung	Langlebige Sessions für UX
HttpOnly	Ja	XSS-Schutz

Eigenschaft	Wert	Sicherheitsrelevanz
Secure	Ja (Production)	MITM-Schutz
SameSite	Strict	CSRF-Schutz

Code-Referenz: Session-Erstellung

```
// application/session/usecases.go:31
type Login func(id ID, login user.Email, password user.Password) (bool, error)
```

7.2.2. Passwort-Authentifizierung

Passwörter werden mit **Argon2id** gehasht, dem von OWASP empfohlenen Algorithmus:

Parameter	Wert	OWASP-Empfehlung
Algorithmus	Argon2id	Empfohlen
Memory	19 MiB	Minimum 19 MiB
Iterationen	2	Minimum 2
Parallelität	1	Minimum 1
Schlüssellänge	32 Byte	Standard

Code-Referenz: Argon2id-Parameter

```
// application/user/password.go:111-113
// OWASP Settings:
https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
// Use Argon2id with minimum configuration of 19 MiB of memory, iteration count of 2,
and 1 degree of parallelism.
func argon2idMin(password string, salt []byte) []byte {
    return argon2.IDKey([]byte(password), salt, 2, 19*1024, 1, 32)
}
```

7.2.3. Passwort-Stärke-Validierung

Die Passwort-Validierung folgt BSI- und OWASP-Empfehlungen:

Kriterium	Anforderung	Status
Minimale Länge	8 Zeichen	Implementiert
Maximale Länge	1000 Zeichen	DoS-Schutz
Sonderzeichen	Mindestens 1	Implementiert
Zahlen	Mindestens 1	Implementiert
Groß-/Kleinschreibung	Beide erforderlich	Implementiert
Entropie-Score	> 60 Bits	Implementiert

Code-Referenz: Passwort-Validierung

```
// application/user/password.go:156-205
func CalculatePasswordStrength(p string) PasswordStrengthIndicator {
    var res PasswordStrengthIndicator
    res.MinLengthRequired = 8
    res.MaxLengthRequired = 1000  // DoS-Schutz
    // ...
    res.Acceptable = res.ContainsMinLength && res.ContainsSpecial &&
                     res.ContainsBelowMaxLength && res.ContainsNumber &&
                     res.ContainsUpperAndLowercase && res.Complexity > Weak
}
```

Security Note aus `application/user/password.go:49`:

> "security note: we must not introduce another sleep here, because argon2id should be safe enough and we must not stall any locks here. We need a totally different attack mitigation strategy."

Argon2id bietet durch seinen Memory-Hard-Charakter ausreichenden Schutz gegen Timing-Attacken.

7.2.4. E-Mail-Verifikation

Die E-Mail-Verifikation ist **verpflichtend** für die Anmeldung:

Code-Referenz: E-Mail-Verifikationspflicht

```
// application/user/uc_authenticates_by_password.go:42-47
if !usr.EMailVerified {
    // security note: intentionally it is not safe to let the user login, if his EMail
was never
    // verified. This opens up all kinds of identity stealing by default, even though
this may
    // be common in the world of shopping systems
    return std.None[User](), std.NewLocalizedError("Login nicht möglich",
        "Das Konto muss zuerst bestätigt werden").WithError(EMailNotVerifiedErr)
}
```

7.2.5. Single Sign-On (SSO)

SSO wird über den **Nago Login Service (NLS)** bereitgestellt:

```
@startuml
title SSO Authentication Flow
```

```
actor User
participant "Nago App" as App
participant "NLS Provider" as NLS

User -> App: Login-Anfrage
App -> App: StartNLSFlow()
App -> NLS: OAuth2 Authorization Request
NLS -> User: Login-Formular
User -> NLS: Credentials
NLS -> App: Authorization Code
App -> App: ExchangeNLS()
App -> NLS: Token Exchange
NLS -> App: Access + Refresh Token
App -> App: MergeSingleSignOnUser()
App -> User: Authenticated Session

@enduml
```

7.2.6. Bootstrap-Admin-Mechanismus

Für initiale Systemkonfiguration existiert ein temporärer Admin-Zugang:

Code-Referenz: Bootstrap Admin mit Zeitlimit

```
// application/management_user.go (via EnableBootstrapAdmin)
// Der Bootstrap-Admin ist zeitlich begrenzt und wird nach Ablauf automatisch
deaktiviert.
// Alle Aktionen werden im Audit-Log erfasst.
```

Der Bootstrap-Admin sollte nur für die initiale Konfiguration verwendet werden. Nach der Einrichtung regulärer Administratoren sollte dieser deaktiviert werden.

7.3. Autorisierung

7.3.1. Role-Based Access Control (RBAC)

Das RBAC-System basiert auf drei Ebenen:

```
@startuml
title RBAC-Modell

class User {
    +ID: string
    +Roles: []RoleID
    +Groups: []GroupID
}

class Role {
```

```
    +ID: string
    +Name: string
    +Permissions: []PermissionID
}

class Permission {
    +ID: string
    +Description: string
}

class Group {
    +ID: string
    +Name: string
    +Roles: []RoleID
}

User "1" -- "*" Role : hat
User "1" -- "*" Group : Mitglied von
Group "1" -- "*" Role : gewährt
Role "1" -- "*" Permission : enthält

@enduml
```

7.3.2. Permission-Audit

Jeder Berechtigungszugriff wird über die `Audit`-Funktion geprüft:

Code-Referenz: Permission-Audit

```
// application/permission/auditable.go
type Auditable interface {
    Audit(permission ID) error
    HasPermission(permission ID) bool
    AuditResource(name string, id string, p ID) error
    HasResourcePermission(name string, id string, p ID) bool
}
```

7.3.3. Privilege Escalation Schutz

Security Note aus `application/flow/evt_repository_assigned.go:27`:

> "security note: this is very important to keep: without this check, an attacker with only flow privileges can escalate privileges by assigning repositories with names of the system (like user, roles, groups) and mutate them in arbitrary ways."

Repository-IDs dürfen nicht mit `nago.` beginnen, um Privilege Escalation zu

> verhindern.

Code-Referenz: Repository-ID-Validierung

```
// application/flow/evt_repository_assigned.go:20-31
func (id RepositoryID) Validate() error {
    s := string(id)
    if strings.HasPrefix(s, "nago.") {
        // Schutz gegen Privilege Escalation
        return fmt.Errorf("repository id cannot start with 'nago.'")
    }
    // ...
}
```

7.4. Kryptographie

7.4.1. Verschlüsselungsarchitektur

```
@startuml
title Kryptographische Schichten

package "Application Layer" {
    [Password Hashing] as PH
    [Session Tokens] as ST
    [API Tokens] as AT
}

package "Storage Layer" {
    [Encrypted Blob Store] as EBS
    [Plain Blob Store] as PBS
}

package "Key Management" {
    [Master Key] as MK
    [Per-Entry Nonces] as PN
}

PH --> [Argon2id]
ST --> [Random Generation]
AT --> [Argon2id without Salt]

EBS --> MK
EBS --> PN
MK --> [AES-GCM-256]
PN --> [AES-GCM-256]

@enduml
```

7.4.2. AES-GCM-256 Implementierung

Code-Referenz: Verschlüsselungsalgorithmus

```
// pkg/blob/crypto/crypto.go:34-56
// Provides symmetric authenticated encryption using 256-bit AES-GCM with a random
nonce.

func encrypt(plaintext []byte, key *[32]byte) (ciphertext []byte, err error) {
    block, err := aes.NewCipher(key[:])
    if err != nil {
        return nil, err
    }

    gcm, err := cipher.NewGCM(block)
    if err != nil {
        return nil, err
    }

    nonce := make([]byte, gcm.NonceSize())
    _, err = io.ReadFull(rand.Reader, nonce)
    if err != nil {
        return nil, err
    }

    return gcm.Seal(nonce, nonce, plaintext, nil), nil
}
```

7.4.3. Master Key Management

Der Master Key ist ein 32-Byte (256-Bit) kryptographischer Schlüssel:

Quelle	Konfiguration	Sicherheit
Umgebungsvariable	`NAGO_MASTER_KEY` (Hex-encoded)	✅ Empfohlen für Production
Lokale Datei	`.masterkey` mit 0600 Permissions	⚠️ Fallback, nicht für Production

Code-Referenz: Master Key Resolution

```
// application/security.go:27-67
func (c *Configurator) MasterKey() (crypto.EncryptionKey, error) {
    hexKey := os.Getenv(envNagoMasterKey)
    if hexKey := strings.TrimSpace(hexKey); hexKey != "" {
        // Aus Umgebungsvariable laden
        buf, err := hex.DecodeString(hexKey)
        // ...
    }

    // Fallback: Datei mit 0600 Permissions
```

```
    if err := os.WriteFile(c.masterKeyFile(),
        []byte(hex.EncodeToString((*key)[:])), 0600); err != nil {
        // ...
    }
}
```

Der Master Key muss **separat** vom Backup gespeichert werden. Bei Verlust des Master Keys können verschlüsselte Daten nicht wiederhergestellt werden.

7.5. Input Validation

7.5.1. User Enumeration Protection

Security Note aus `application/management_mail.go:126`:

> "security note: intentionally do not expose this information to the frontend"

Bei Password-Reset-Anfragen wird nicht offengelegt, ob der Benutzer existiert.

Code-Referenz: User Enumeration Schutz

```
// application/management_mail.go:125-129
if optUser.IsNone() {
    // security note: intentionally do not expose this information to the frontend
    slog.Error("shall send verification mail but user not found", "mail", mail)
    return nil  // Kein Fehler zurückgeben
}
```

7.5.2. Bewusste User-Existenz-Offenlegung bei Registrierung

Security Note aus `application/user/uc_create.go:130`:

> "security note: this allows to expose the fact, that a user already exists in the system. however, there is no reasonable way to avoid that. We must not delay this with a sleep, because we are still in the write mutex lock and therefore would cause a kind of accumulating 'deadlock'/DOS like behavior"

Bei der Registrierung wird bewusst offengelegt, wenn eine E-Mail bereits existiert. Dies ist ein akzeptierter Trade-off zwischen UX und Security.

7.5.3. DoS-Schutzmaßnahmen

Mutex-Lock für User-Erstellung

Code-Referenz: Rate-Limiting bei User-Erstellung

```
// application/user/uc_create.go:31-32
// this is really harsh and allows intentionally only to create one user per second
mutex.Lock()
defer mutex.Unlock()
```

Passwort-Längenbegrenzung

Code-Referenz: DoS-Schutz durch maximale Passwortlänge

```
// application/user/password.go:167-170
// see
https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
if strlen <= res.MaxLengthRequired {  // 1000 Zeichen
    // probably a DOS attack
    res.ContainsBelowMaxLength = true
}
```

7.5.4. Staging-ID-Sicherheit

Security Note aus application/dataimport/ui/page_select_parser.go:72:

> "security note: do not extend this by accepting existing staging id by query because our use cases have no security concept, which otherwise protects against inserting malicious data in foreign staging data sets."

Staging-IDs werden nicht via Query-Parameter akzeptiert, um Dateninjection zu verhindern.

7.6. Netzwerksicherheit

7.6.1. CORS-Konfiguration

Code-Referenz: CORS im Debug-Modus

```
// application/cfg_page.go:206-217
if c.debug {
    r.Use(
        cors.Handler(cors.Options{
            AllowedOrigins:   []string{"http://*"},  // NUR DEBUG!
            AllowedMethods:   []string{"GET", "POST", "PUT", "DELETE", "OPTIONS"},
```

```
            AllowedHeaders:   []string{"Accept", "Authorization", "Content-Type", "X-
CSRF-Token"},
            ExposedHeaders:   []string{"Link"},
            AllowCredentials: true,
            MaxAge:           300,
        }),
    )
    c.defaultLogger().Warn("using debug cors settings")
}
```

Die Debug-CORS-Einstellungen (`AllowedOrigins: ["http://"])` dürfen *niemals* in Production verwendet werden!

7.6.2. WebSocket Security

Code-Referenz: WebSocket Origin-Validierung

```
// application/cfg_page.go:507-510
var upgrader = websocket.Upgrader{
    CheckOrigin: func(r *http.Request) bool {
        return true //TODO security implications?
    },
    EnableCompression: true,
}
```

TODO: Die WebSocket Origin-Validierung ist aktuell deaktiviert (`CheckOrigin: true`). Dies sollte für Production mit einer strikten Origin-Prüfung implementiert werden.

7.6.3. TLS für SMTP

Code-Referenz: TLS-Konfiguration für SMTP

```
// application/mail/smtp.go:28-31
tlsconfig := &tls.Config{
    InsecureSkipVerify: false,  // Zertifikatsvalidierung aktiviert
    ServerName:         host,
}
```

7.7. Logging und Audit

7.7.1. Strukturiertes Logging

Das Framework verwendet Go's `slog` für strukturiertes Logging:

```
slog.Info("user created",
```

```
    slog.String("userID", user.ID),
    slog.String("email", user.Email),
    slog.Time("createdAt", user.CreatedAt))
```

7.7.2. Event-basierter Audit Trail

Alle sicherheitsrelevanten Ereignisse werden über den Event Bus publiziert:

Code-Referenz: User-Created Event

```
// application/user/uc_create.go:157-166
eventBus.Publish(Created{
    ID:                user.ID,
    Firstname:         user.Contact.Firstname,
    Lastname:          user.Contact.Lastname,
    Email:             user.Email,
    PreferredLanguage: tag,
    NotifyUser:        notify,
    VerificationCode:  user.VerificationCode,
    CreatedAt:         user.CreatedAt,
})
```

7.7.3. Audit-UI

Die Audit-Funktionalität ist über das Admin Center zugänglich:

Code-Referenz: Audit Page

```
// application/evs/ui/page_audit.go:35-44
func PageAudit[Evt any](wnd core.Window, uc evs.UseCases[Evt], opts PageAuditOptions
[Evt]) core.View {
    // Zeigt alle Events mit Zeitstempel, User, Aktion
}
```

8. ISO 27001 Annex A Controls Mapping

8.1. Übersicht

Dieses Kapitel ordnet die implementierten Sicherheitsmaßnahmen des Nago Frameworks den Kontrollen des ISO 27001:2022 Annex A zu.

8.2. Statement of Applicability (SoA)

Control ID	Beschreibung	Status	Implementierung
A.5 Organisatorische Kontrollen			

Control ID	Beschreibung	Status	Implementierung
A.5.15	Zugriffskontrolle	✅ Implementiert	RBAC-System, Permission-basierte Kontrollen
A.5.16	Identitätsmanagement	✅ Implementiert	User Management mit E-Mail-Verifikation
A.5.17	Authentifizierungsinformationen	✅ Implementiert	Argon2id Passwort-Hashing, SSO via NLS
A.8 Technologische Kontrollen			
A.8.2	Privilegierte Zugriffsrechte	✅ Implementiert	Permission System mit Audit-Trail
A.8.3	Einschränkung des Informationszugriffs	✅ Implementiert	`subject.Audit(permission)` Prüfung
A.8.4	Zugang zu Quellcode	✅ Implementiert	Repository-ID-Validierung gegen Escalation
A.8.5	Sichere Authentifizierung	✅ Implementiert	Argon2id, E-Mail-Verifikation Pflicht
A.8.9	Konfigurationsmanagement	✅ Implementiert	Master Key via Env/File, CORS-Konfiguration
A.8.12	Verhinderung von Datenlecks	✅ Implementiert	Verschlüsselung mit AES-GCM-256
A.8.13	Informationssicherung	✅ Implementiert	Backup-System mit Verschlüsselung
A.8.16	Überwachungsaktivitäten	✅ Implementiert	Event Bus, Structured Logging mit slog
A.8.24	Verwendung von Kryptographie	✅ Implementiert	AES-GCM-256, Argon2id, TLS
A.8.25	Sichere Entwicklung	✅ Implementiert	Security Notes im Code, Input Validation

8.3. Detaillierte Control-Evidenzen

8.3.1. A.5.15 - Zugriffskontrolle

Anforderung: Regeln zur Kontrolle des physischen und logischen Zugangs zu Informationen und anderen zugehörigen Vermögenswerten sollen auf der Grundlage von Geschäfts- und Informationssicherheitsanforderungen festgelegt und umgesetzt werden.

Implementierung:

```
// application/permission/auditable.go
type Auditable interface {
    // Audit prüft ob die Permission vorhanden ist und gibt einen Fehler zurück
    Audit(permission ID) error
    // HasPermission prüft ob die Permission vorhanden ist
    HasPermission(permission ID) bool
    // AuditResource prüft ressourcenspezifische Berechtigungen
    AuditResource(name string, id string, p ID) error
}
```

Evidenz:

- Datei: `application/permission/auditable.go`
- Alle Use Cases prüfen Berechtigungen vor Ausführung
- Audit-Trail über Event Bus

8.3.2. A.5.16 - Identitätsmanagement

Anforderung: Der gesamte Lebenszyklus von Identitäten soll verwaltet werden.

Implementierung:

```
// application/user/usecases.go
type UseCases struct {
    FindByID               FindByID
    FindByMail             FindByMail
    Create                 Create          // Benutzer erstellen
    Update                 Update          // Benutzer aktualisieren
    Delete                 Delete          // Benutzer löschen
    ChangeMyPassword       ChangeMyPassword
    ResetPasswordRequestCode ResetPasswordRequestCode
    EnableBootstrapAdmin   EnableBootstrapAdmin
}
```

Evidenz:

- Datei: `application/user/usecases.go`

- Vollständiger CRUD-Lebenszyklus für Benutzer
- E-Mail-Verifikation als Pflicht

8.3.3. A.5.17 - Authentifizierungsinformationen

Anforderung: Die Zuweisung und Verwaltung von Authentifizierungsinformationen soll durch einen Verwaltungsprozess kontrolliert werden.

Implementierung:

```
// application/user/password.go:111-113
// OWASP-konforme Passwort-Speicherung mit Argon2id
func argon2idMin(password string, salt []byte) []byte {
    return argon2.IDKey([]byte(password), salt, 2, 19*1024, 1, 32)
}
```

Evidenz:

- Datei: `application/user/password.go:111-113`
- Argon2id mit OWASP-Minimalparametern
- Passwort-Stärke-Validierung nach BSI-Empfehlungen

8.3.4. A.8.2 - Privilegierte Zugriffsrechte

Anforderung: Die Zuweisung und Nutzung von privilegierten Zugriffsrechten soll eingeschränkt und verwaltet werden.

Implementierung:

```
// application/flow/evt_repository_assigned.go:27-29
// security note: Schutz gegen Privilege Escalation
if strings.HasPrefix(s, "nago.") {
    return fmt.Errorf("repository id cannot start with 'nago.'")
}
```

Evidenz:

- Datei: `application/flow/evt_repository_assigned.go:27`
- System-Repositories sind gegen Manipulation geschützt
- Privilege Escalation wird verhindert

8.3.5. A.8.5 - Sichere Authentifizierung

Anforderung: Sichere Authentifizierungstechnologien und -verfahren sollen auf der Grundlage von Einschränkungen des Informationszugriffs und der themenspezifischen Richtlinie zur Zugriffskontrolle umgesetzt werden.

Implementierung:

```
// application/user/uc_authenticates_by_password.go:42-47
if !usr.EMailVerified {
    // security note: E-Mail-Verifikation ist Pflicht
    return std.None[User](), std.NewLocalizedError("Login nicht möglich",
        "Das Konto muss zuerst bestätigt werden").WithError(EMailNotVerifiedErr)
}
```

Evidenz:

- Datei: `application/user/uc_authenticates_by_password.go:42-47`
- E-Mail-Verifikation als Sicherheitsgate
- Schutz gegen Identity Stealing

8.3.6. A.8.12 - Verhinderung von Datenlecks

Anforderung: Maßnahmen zur Verhinderung von Datenlecks sollen auf Systeme, Netzwerke und alle anderen Geräte angewendet werden, die sensible Informationen verarbeiten, speichern oder übertragen.

Implementierung:

```
// application/management_mail.go:125-129
if optUser.IsNone() {
    // security note: intentionally do not expose this information
    slog.Error("shall send verification mail but user not found", "mail", mail)
    return nil  // Kein Fehler zurückgeben um User Enumeration zu verhindern
}
```

Evidenz:

- Datei: `application/management_mail.go:126`
- User Enumeration wird verhindert
- Keine Offenlegung von Benutzerexistenz bei Password-Reset

8.3.7. A.8.13 - Informationssicherung

Anforderung: Backup-Kopien von Informationen, Software und Systemen sollen gemäß der vereinbarten themenspezifischen Richtlinie zur Datensicherung aufbewahrt und regelmäßig getestet werden.

Implementierung:

```
// application/backup/usecases.go
type UseCases struct {
```

```
    Backup           Backup           // Vollständiges Backup erstellen
    Restore          Restore          // Aus Backup wiederherstellen
    ExportMasterKey  ExportMasterKey  // Master Key exportieren
    ReplaceMasterKey ReplaceMasterKey // Master Key ersetzen
}
```

Evidenz:

- Datei: `application/backup/usecases.go`
- Verschlüsselte Stores bleiben verschlüsselt im Backup
- Master Key separat verwaltet

8.3.8. A.8.16 - Überwachungsaktivitäten

Anforderung: Netzwerke, Systeme und Anwendungen sollen auf anomales Verhalten überwacht werden, und es sollen geeignete Maßnahmen ergriffen werden, um potenzielle Informationssicherheitsvorfälle zu bewerten.

Implementierung:

```
// application/evs/ui/page_audit.go
func PageAudit[Evt any](wnd core.Window, uc evs.UseCases[Evt],
    opts PageAuditOptions[Evt]) core.View {
    // Zeigt alle Events mit Zeitstempel, User, Aktion
}
```

Evidenz:

- Datei: `application/evs/ui/page_audit.go`
- Event-basierter Audit Trail
- Strukturiertes Logging mit slog

8.3.9. A.8.24 - Verwendung von Kryptographie

Anforderung: Regeln für die effektive Nutzung von Kryptographie, einschließlich der Verwaltung kryptografischer Schlüssel, sollen definiert und umgesetzt werden.

Implementierung:

```
// pkg/blob/crypto/crypto.go:34-56
// AES-GCM-256 mit random Nonce
func encrypt(plaintext []byte, key *[32]byte) (ciphertext []byte, err error) {
    block, err := aes.NewCipher(key[:])
    gcm, err := cipher.NewGCM(block)
    nonce := make([]byte, gcm.NonceSize())
    io.ReadFull(rand.Reader, nonce)
    return gcm.Seal(nonce, nonce, plaintext, nil), nil
```

```
}
```

Evidenz:

- Datei: `pkg/blob/crypto/crypto.go:34-56`
- AES-GCM-256 für Datenverschlüsselung
- Argon2id für Passwort-Hashing
- TLS für Netzwerkkommunikation

8.3.10. A.8.25 - Sichere Entwicklung

Anforderung: Regeln für die sichere Entwicklung von Software und Systemen sollen festgelegt und angewendet werden.

Implementierung:

Das Framework enthält **20 dokumentierte Security Notes** im Quellcode:

Datei	Security Note
`flow/evt_repository_assigned.go:27`	Privilege Escalation Schutz durch Repository-ID-Validierung
`session/ui/login.go:145`	User Enumeration Schutz bei Login-Fehlern
`management_mail.go:126,169`	Keine User-Existenz-Offenlegung bei Password-Reset
`management_mail.go:131,174`	Neue Security Codes bei jedem Reset
`user/password.go:49`	Keine zusätzlichen Sleeps wegen Argon2id-Sicherheit
`user/password.go:52`	Trade-off bei schwachen bestehenden Passwörtern
`dataimport/ui/page_select_parser.go:72`	Keine Staging-ID via Query-Parameter
`usercircle/ui/view_users.go:32`	UI-Schutz durch Circle-Admin-Prüfung

Evidenz:

- Security Notes sind im Quellcode dokumentiert
- Jede Entscheidung ist nachvollziehbar begründet
- Code Reviews berücksichtigen Security-Aspekte

8.4. Compliance-Matrix

Control	Applicable	Implemented	Justification
A.5.15	Ja	Ja	RBAC mit Permission-Audit
A.5.16	Ja	Ja	User Management Lifecycle
A.5.17	Ja	Ja	Argon2id, E-Mail-Verifikation

Control	Applicable	Implemented	Justification
A.8.2	Ja	Ja	Repository-ID-Validierung
A.8.3	Ja	Ja	subject.Audit() Prüfung
A.8.4	Ja	Ja	nago.* Präfix-Schutz
A.8.5	Ja	Ja	E-Mail-Verifikation Pflicht
A.8.9	Ja	Ja	Env/File Konfiguration
A.8.12	Ja	Ja	User Enumeration Schutz
A.8.13	Ja	Ja	Verschlüsselte Backups
A.8.16	Ja	Ja	Event Bus Audit Trail
A.8.24	Ja	Ja	AES-GCM-256, Argon2id, TLS
A.8.25	Ja	Ja	Security Notes im Code

9. STRIDE Bedrohungsmodell

9.1. Übersicht

Diese Analyse verwendet das STRIDE-Modell zur systematischen Identifikation von Bedrohungen für das Nago Framework. Jede Bedrohungskategorie wird mit implementierten Mitigationen dokumentiert.

9.2. STRIDE-Kategorien

```
@startuml
title STRIDE Bedrohungsmodell - Nago Framework

rectangle "Bedrohungen" {
    rectangle "Spoofing" as S #ffcccc
    rectangle "Tampering" as T #ffddcc
    rectangle "Repudiation" as R #ffffcc
    rectangle "Information\nDisclosure" as I #ccffcc
    rectangle "Denial of\nService" as D #ccccff
    rectangle "Elevation of\nPrivilege" as E #ffccff
}

rectangle "Komponenten" {
    rectangle "Session\nManagement" as SM
    rectangle "User\nManagement" as UM
    rectangle "Permission\nSystem" as PM
    rectangle "Blob\nStorage" as BS
    rectangle "Backup\nSystem" as BU
    rectangle "WebSocket" as WS
}
```

```
S --> SM : Session Hijacking
S --> UM : Identity Theft
T --> BS : Data Manipulation
T --> BU : Backup Tampering
R --> PM : Audit Evasion
I --> BS : Data Leakage
I --> BU : Master Key Exposure
D --> UM : Brute Force
D --> WS : Connection Flooding
E --> PM : Permission Bypass
E --> UM : Privilege Escalation

@enduml
```

9.3. Spoofing (Identitätsverschleierung)

9.3.1. S-01: Session Hijacking

Bedrohung	**Angreifer stiehlt Session-Cookie und übernimmt Benutzersitzung**
Wahrscheinlichkeit	Mittel
Auswirkung	Hoch - Vollständiger Zugriff auf Benutzerkonto
Mitigationen	• HttpOnly Cookie verhindert JavaScript-Zugriff • Secure Flag erzwingt HTTPS-Übertragung • SameSite=Strict verhindert CSRF • Session-TTL von 3 Monaten begrenzt Exposure-Fenster
Status	Mitigiert

9.3.2. S-02: Token Theft via API

Bedrohung	**Angreifer extrahiert API-Token und authentifiziert sich**
Wahrscheinlichkeit	Niedrig
Auswirkung	Hoch - API-Zugriff mit Token-Berechtigungen
Mitigationen	• API-Tokens werden mit Argon2id gehasht gespeichert • Reverse Hash Lookup ohne Salt (Performance-Trade-off) • Token-Rotation möglich
Code-Referenz	`application/user/password.go:97-108`
Status	Mitigiert

9.3.3. S-03: User Enumeration bei Login

Bedrohung	**Angreifer erkennt gültige Benutzerkonten durch unterschiedliche Fehlermeldungen**
Wahrscheinlichkeit	Mittel
Auswirkung	Niedrig - Information für Brute-Force-Vorbereitung
Mitigationen	• Einheitliche Fehlermeldung: "Der Benutzer existiert nicht, das Konto wurde deaktiviert oder das Kennwort ist falsch." • Security Note in `session/ui/login.go:145`
Code-Referenz	`application/session/ui/login.go:145`
Status	✅ Mitigiert

9.4. Tampering (Datenmanipulation)

9.4.1. T-01: MITM-Angriff auf Kommunikation

Bedrohung	**Angreifer manipuliert Daten während der Übertragung**
Wahrscheinlichkeit	Niedrig (mit TLS)
Auswirkung	Hoch - Datenintegrität kompromittiert
Mitigationen	• TLS für alle HTTP-Verbindungen (Deployment-Ebene) • TLS für SMTP mit Zertifikatsvalidierung • WebSocket über WSS
Code-Referenz	`application/mail/smtp.go:28-31`
Status	✅ Mitigiert (Deployment-abhängig)

9.4.2. T-02: Backup-Manipulation

Bedrohung	**Angreifer manipuliert Backup-Datei und führt kompromittierte Daten ein**
Wahrscheinlichkeit	Niedrig
Auswirkung	Kritisch - Systemübernahme möglich
Mitigationen	• Verschlüsselte Stores bleiben verschlüsselt • Master Key separat gespeichert • Backup-Restore nur mit Berechtigung
Code-Referenz	`application/backup/usecases.go`
Status	⚠️ Teilweise mitigiert - Keine kryptographische Signatur

9.4.3. T-03: Repository-Daten-Manipulation

Bedrohung	**Angreifer mit Flow-Rechten manipuliert System-Repositories**
Wahrscheinlichkeit	Mittel
Auswirkung	Kritisch - Privilege Escalation
Mitigationen	• Repository-IDs dürfen nicht mit `nago.` beginnen • Validierung bei Zuweisung
Code-Referenz	`application/flow/evt_repository_assigned.go:27`
Status	☐ Mitigiert

9.5. Repudiation (Abstreitbarkeit)

9.5.1. R-01: Audit-Log-Manipulation

Bedrohung	**Angreifer löscht oder manipuliert Audit-Logs**
Wahrscheinlichkeit	Niedrig
Auswirkung	Hoch - Forensik nicht möglich
Mitigationen	• Event Store als Append-Only-Struktur • Events enthalten Zeitstempel und User-ID • Zugriff auf Event Store nur mit Berechtigung
Code-Referenz	`application/evs/`
Status	☐ Mitigiert

9.5.2. R-02: Aktions-Abstreitbarkeit

Bedrohung	**Benutzer bestreitet durchgeführte Aktion**
Wahrscheinlichkeit	Mittel
Auswirkung	Mittel - Compliance-Probleme
Mitigationen	• Alle sicherheitsrelevanten Aktionen werden geloggt • Event-Envelope enthält User-ID, Zeitstempel, Aktion • Strukturiertes Logging mit slog
Code-Referenz	`application/evs/ui/page_audit.go`
Status	☐ Mitigiert

9.6. Information Disclosure (Informationsoffenlegung)

9.6.1. I-01: Master Key Exposure

Bedrohung	Angreifer erhält Zugriff auf Master Key
Wahrscheinlichkeit	Niedrig
Auswirkung	Kritisch - Alle verschlüsselten Daten kompromittiert
Mitigationen	• Umgebungsvariable `NAGO_MASTER_KEY` (empfohlen) • Datei `.masterkey` mit 0600 Permissions (Fallback) • Master Key nicht im Backup enthalten
Code-Referenz	`application/security.go:27-67`
Status	Mitigiert

9.6.2. I-02: User Enumeration bei Password-Reset

Bedrohung	Angreifer erkennt gültige E-Mail-Adressen
Wahrscheinlichkeit	Mittel
Auswirkung	Niedrig - Information für Phishing
Mitigationen	• Keine Unterscheidung ob User existiert oder nicht • Gleiche Antwort bei existierendem und nicht-existierendem User
Code-Referenz	`application/management_mail.go:126`
Status	Mitigiert

9.6.3. I-03: User Existenz bei Registrierung

Bedrohung	Angreifer erkennt registrierte E-Mail-Adressen
Wahrscheinlichkeit	Hoch
Auswirkung	Niedrig - Akzeptierter Trade-off
Mitigationen	• Bewusste Design-Entscheidung dokumentiert • Keine Sleep-Verzögerung wegen Mutex-Lock • Alternative: Captcha (nicht implementiert)
Code-Referenz	`application/user/uc_create.go:130`
Status	Akzeptiertes Risiko

9.6.4. I-04: Backup-Datei-Leak

Bedrohung	Backup-Datei wird unbefugt zugänglich
Wahrscheinlichkeit	Niedrig
Auswirkung	Hoch - Sensible Daten exponiert

Bedrohung	Backup-Datei wird unbefugt zugänglich
Mitigationen	• Verschlüsselte Stores bleiben verschlüsselt • Master Key separat gespeichert • Unverschlüsselte Daten: Risikoanalyse erforderlich
Code-Referenz	`application/backup/usecases.go`
Status	⚠ Teilweise mitigiert

9.7. Denial of Service (Dienstverweigerung)

9.7.1. D-01: Brute-Force-Login

Bedrohung	Angreifer versucht Passwörter durch Massenanfragen zu erraten
Wahrscheinlichkeit	Hoch
Auswirkung	Mittel - Kontoübernahme bei schwachem Passwort
Mitigationen	• Argon2id macht Brute-Force teuer (Memory-Hard) • Passwort-Stärke-Validierung erzwingt starke Passwörter • ⚠ Kein globales Rate-Limiting implementiert
Code-Referenz	`application/user/password.go:111`
Status	⚠ Teilweise mitigiert - Rate-Limiting fehlt

9.7.2. D-02: User-Erstellung-DoS

Bedrohung	Angreifer überflutet System mit Registrierungsanfragen
Wahrscheinlichkeit	Mittel
Auswirkung	Mittel - System wird langsam
Mitigationen	• Mutex-Lock erlaubt nur 1 User/Sekunde • E-Mail-Verifikation verhindert Spam-Accounts
Code-Referenz	`application/user/uc_create.go:31-32`
Status	✓ Mitigiert

9.7.3. D-03: Password-Hashing-DoS

Bedrohung	Angreifer sendet extrem lange Passwörter
Wahrscheinlichkeit	Mittel
Auswirkung	Mittel - CPU/Memory-Erschöpfung
Mitigationen	• Maximale Passwortlänge: 1000 Zeichen • Validierung vor Hashing

Bedrohung	**Angreifer sendet extrem lange Passwörter**
Code-Referenz	`application/user/password.go:167`
Status	✅ Mitigiert

9.7.4. D-04: WebSocket Connection Flooding

Bedrohung	**Angreifer öffnet viele WebSocket-Verbindungen**
Wahrscheinlichkeit	Mittel
Auswirkung	Hoch - Server-Ressourcen erschöpft
Mitigationen	• ⚠️ Keine Verbindungslimits implementiert • Deployment-Ebene: Reverse Proxy Limits
Status	⚠️ Nicht mitigiert auf Application-Ebene

9.8. Elevation of Privilege (Rechteausweitung)

9.8.1. E-01: Permission Bypass

Bedrohung	**Angreifer umgeht Berechtigungsprüfungen**
Wahrscheinlichkeit	Niedrig
Auswirkung	Kritisch - Unbefugter Zugriff
Mitigationen	• Alle Use Cases verwenden `subject.Audit(permission)` • Fehlende Berechtigung gibt Error zurück • Audit-Trail für alle Zugriffe
Code-Referenz	`application/permission/auditable.go`
Status	✅ Mitigiert

9.8.2. E-02: Repository-ID-Escalation

Bedrohung	**Angreifer mit Flow-Rechten übernimmt System-Repositories**
Wahrscheinlichkeit	Mittel
Auswirkung	Kritisch - System-Kompromittierung
Mitigationen	• Repository-IDs dürfen nicht mit `nago.` beginnen • Validierung bei jeder Repository-Zuweisung • Security Note dokumentiert
Code-Referenz	`application/flow/evt_repository_assigned.go:27`
Status	✅ Mitigiert

9.8.3. E-03: Bootstrap-Admin-Missbrauch

Bedrohung	**Bootstrap-Admin wird für persistenten Zugriff missbraucht**
Wahrscheinlichkeit	Niedrig
Auswirkung	Kritisch - Administratorzugriff
Mitigationen	• Zeitlimit für Bootstrap-Admin • Automatische Deaktivierung nach Ablauf • Audit-Log für alle Admin-Aktionen
Code-Referenz	`application/management_user.go`
Status	✅ Mitigiert

9.8.4. E-04: User Circle Bypass

Bedrohung	**Benutzer greift auf Ressourcen außerhalb seines Circles zu**
Wahrscheinlichkeit	Mittel
Auswirkung	Mittel - Unbefugter Ressourcenzugriff
Mitigationen	• UI prüft Circle-Admin-Status bei jedem Render • Sofortiger Exit bei fehlendem Zugriff • Security Note dokumentiert
Code-Referenz	`application/usercircle/ui/view_users.go:32`
Status	✅ Mitigiert

9.9. Risiko-Matrix

```
@startuml
title Risiko-Matrix

scale 600 width

rectangle "Kritisch" as C #ff0000
rectangle "Hoch" as H #ff8800
rectangle "Mittel" as M #ffff00
rectangle "Niedrig" as L #00ff00

note right of C
  E-01, E-02, E-03
  I-01, T-02
end note

note right of H
  S-01, S-02
  D-04, I-04
```

```
end note

note right of M
  S-03, D-01
  E-04, R-02
end note

note right of L
  T-01, R-01
  D-02, D-03
end note

@enduml
```

9.10. Zusammenfassung der offenen Punkte

ID	Bedrohung	Risiko	Empfehlung
D-01	Brute-Force-Login	Mittel	Globales Rate-Limiting implementieren
D-04	WebSocket Flooding	Hoch	Connection Limits auf Application-Ebene
T-02	Backup-Manipulation	Mittel	Kryptographische Signatur für Backups
I-03	User Enumeration bei Registrierung	Niedrig	Captcha oder alternative UX erwägen
WebSocket	Origin-Validierung	Hoch	CheckOrigin-Funktion implementieren (TODO in Code)

10. OWASP Top 10 Pentest-Checkliste

10.1. Übersicht

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

10.2. Legende

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

10.3. A01:2021 - Broken Access Control

10.3.1. 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

10.3.2. 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
```

10.4. A02:2021 - Cryptographic Failures

10.4.1. 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

10.4.2. 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)
```

10.5. A03:2021 - Injection

10.5.1. 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

10.5.2. 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
```

10.6. A04:2021 - Insecure Design

10.6.1. 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

10.6.2. 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!
```

10.7. A05:2021 - Security Misconfiguration

10.7.1. 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

10.7.2. 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)
```

10.8. A06:2021 - Vulnerable and Outdated Components

10.8.1. 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

10.8.2. 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
```

10.9. A07:2021 - Identification and Authentication Failures

10.9.1. 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

10.9.2. 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
```

10.10. A08:2021 - Software and Data Integrity Failures

10.10.1. 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

10.10.2. 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
```

10.11. A09:2021 - Security Logging and Monitoring Failures

10.11.1. 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

10.11.2. 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
```

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

10.12.1. 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`

10.12.2. 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!
```

10.13. Zusammenfassung

10.13.1. Kritische Findings

Kategorie	Finding	Empfehlung
A04	WebSocket Origin-Validierung deaktiviert	CheckOrigin-Funktion implementieren

Kategorie	Finding	Empfehlung
A05	Debug-Mode CORS in Production möglich	Automatische Deaktivierung in Production
A07	Kein globales Rate-Limiting für Login	Rate-Limiter implementieren

10.13.2. 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

10.13.3. 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

11. Compliance-Nachweise

11.1. Übersicht

Dieses Kapitel dokumentiert die Compliance-Nachweise für verschiedene Sicherheitsstandards und Regularien.

11.2. ISO 27001:2022

11.2.1. Statement of Applicability (SoA)

Die folgende Tabelle zeigt die Anwendbarkeit und Implementierung der ISO 27001:2022 Annex A Controls:

Control	Anwendbar	Implementiert	Evidenz
A.5.15	Ja	Ja	`application/permission/auditable.go` - RBAC-System
A.5.16	Ja	Ja	`application/user/usecases.go` - User Lifecycle
A.5.17	Ja	Ja	`application/user/password.go:111` - Argon2id
A.8.2	Ja	Ja	`application/flow/evt_repository_assigned.go:27`
A.8.3	Ja	Ja	`application/permission/auditable.go` - Audit()
A.8.5	Ja	Ja	`application/user/uc_authenticates_by_password.go:42`

Control	Anwendbar	Implementiert	Evidenz
A.8.9	Ja	Ja	`application/security.go` - Master Key Management
A.8.12	Ja	Ja	`application/management_mail.go:126` - User Enumeration
A.8.13	Ja	Ja	`application/backup/usecases.go` - Encrypted Backups
A.8.16	Ja	Ja	`application/evs/` - Event Store Audit Trail
A.8.24	Ja	Ja	`pkg/blob/crypto/crypto.go` - AES-GCM-256
A.8.25	Ja	Ja	20 Security Notes im Quellcode dokumentiert

11.2.2. Audit-Evidenz-Katalog

A.5.17 - Authentifizierungsinformationen

Evidenz-Typ: Quellcode

Datei: `application/user/password.go`

Zeilen: 111-113

```
// OWASP-konforme Passwort-Speicherung
// https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
func argon2idMin(password string, salt []byte) []byte {
    return argon2.IDKey([]byte(password), salt, 2, 19*1024, 1, 32)
}
```

Nachweis: Argon2id mit OWASP-Minimalparametern implementiert.

A.8.5 - Sichere Authentifizierung

Evidenz-Typ: Quellcode

Datei: `application/user/uc_authenticates_by_password.go`

Zeilen: 42-47

```
if !usr.EMailVerified {
    // security note: intentionally it is not safe to let the user login
    return std.None[User](), std.NewLocalizedError("Login nicht möglich",
        "Das Konto muss zuerst bestätigt werden").WithError(EMailNotVerifiedErr)
}
```

Nachweis: E-Mail-Verifikation ist Voraussetzung für Login.

A.8.24 - Verwendung von Kryptographie

Evidenz-Typ: Quellcode

Datei: `pkg/blob/crypto/crypto.go`

Zeilen: 34-56

```
// 256-bit AES-GCM with random nonce
func encrypt(plaintext []byte, key *[32]byte) (ciphertext []byte, err error) {
    block, err := aes.NewCipher(key[:])
    gcm, err := cipher.NewGCM(block)
    nonce := make([]byte, gcm.NonceSize())
    io.ReadFull(rand.Reader, nonce)
    return gcm.Seal(nonce, nonce, plaintext, nil), nil
}
```

Nachweis: AES-GCM-256 mit Random Nonce implementiert.

11.3. SOC 2 Type II

11.3.1. Trust Service Criteria Mapping

Kriterium	Beschreibung	Implementierung	Evidenz
Security (Common Criteria)			
CC6.1	Logische Zugangssicherheit	RBAC mit Permission-System	`application/permission/`
CC6.2	Zugangsbeschränkungen	Permission-Audit vor jeder Aktion	`subject.Audit()`
CC6.3	Registrierung und Autorisierung	User Management mit E-Mail-Verifikation	`application/user/`
CC6.6	Verschlüsselung	AES-GCM-256 für sensible Daten	`pkg/blob/crypto/`
CC6.7	Zugang zu Systemen	Session-basierte Authentifizierung	`application/session/`
Availability			
CC7.1	System-Überwachung	Event Bus Logging	`application/evs/`
CC7.2	Vorfallsreaktion	Audit Trail für Forensik	Event Store
Processing Integrity			
CC8.1	Input-Validierung	Password-Stärke, E-Mail-Format	`application/user/password.go`
Confidentiality			

Kriterium	Beschreibung	Implementierung	Evidenz
CC9.1	Vertrauliche Informationen	Encrypted Blob Storage	`pkg/blob/crypto/`
CC9.2	Entsorgung	Backup mit verschlüsselten Stores	`application/backup/`

11.4. DSGVO/GDPR Compliance

11.4.1. Artikel-Mapping

Artikel	Anforderung	Implementierung
Art. 5 (1) f	Integrität und Vertraulichkeit	AES-GCM-256 Verschlüsselung, TLS, Argon2id
Art. 25	Privacy by Design	E-Mail-Verifikation Pflicht, Verschlüsselung Standard
Art. 32 (1) a	Pseudonymisierung und Verschlüsselung	Master Key Verschlüsselung für sensible Stores
Art. 32 (1) b	Vertraulichkeit, Integrität	RBAC, Permission-Audit, AES-GCM Authentication Tag
Art. 32 (1) c	Belastbarkeit	Session-Recovery, Backup-Restore
Art. 32 (1) d	Überprüfungsverfahren	Audit Trail, Structured Logging
Art. 33	Meldung von Verletzungen	Siehe Incident Response Plan
Art. 34	Benachrichtigung Betroffener	Siehe Incident Response Plan

11.4.2. Consent-Verwaltung

Das Framework unterstützt DSGVO-konforme Einwilligungsverwaltung:

```
// example/cmd/tutorial-62-dataimporter/main.go:61-80
usrSettings.Consents = []user.ConsentOption{
    {
        ID:       consent.DataProtectionProvision,
        Register: user.ConsentText{
            Label: "Ja, ich habe die [Datenschutzbestimmungen](/page/datenschutz)
gelesen und akzeptiert."
        },
        Required: true,
    },
    {
        ID: consent.Newsletter,
        Register: user.ConsentText{
            Label: "Ja, ich melde mich zum Newsletter an.",
            SupportingText: "Ein Widerspruch ist jederzeit möglich.",
        },
```

```
    },
}
```

Evidenz:

- Pflicht-Consent für Datenschutzbestimmungen
- Optionaler Newsletter-Consent
- Widerrufsmöglichkeit dokumentiert

11.5. BSI IT-Grundschutz

11.5.1. Baustein-Mapping

Baustein	Anforderung	Implementierung
APP.3.1	Webanwendungen und Webservices	Input Validation, XSS-Schutz via VueJS
APP.3.2	Webserver	chi Router mit Middleware, Structured Logging
CON.1	Kryptokonzept	AES-GCM-256, Argon2id, TLS
CON.3	Datensicherungskonzept	Backup-System mit Verschlüsselung
OPS.1.1.2	Ordnungsgemäße IT-Administration	RBAC, Audit Trail, Bootstrap-Admin mit Zeitlimit
OPS.1.2.5	Fernwartung	SSO via NLS, Session-basierte Authentifizierung
DER.1	Detektion von sicherheitsrelevanten Ereignissen	Event Bus, Structured Logging, Audit-UI

11.6. Penetrationstest-Historie

11.6.1. Template für Pentest-Berichte

Feld	Beschreibung
Testzeitraum	[Datum von - bis]
Tester	[Name/Firma]
Scope	Nago Framework Application Layer
Methodik	OWASP Testing Guide, STRIDE
Findings	[Anzahl Kritisch/Hoch/Mittel/Niedrig]
Status	[Offen/In Bearbeitung/Geschlossen]

11.6.2. Finding-Template

Feld	Wert
ID	[PT-YYYY-NNN]
Titel	[Kurzbeschreibung]
Schweregrad	[Kritisch/Hoch/Mittel/Niedrig]
CVSS Score	[0.0-10.0]
Betroffene Komponente	[Pfad/Datei]
Beschreibung	[Detaillierte Beschreibung]
Reproduktion	[Schritte zur Reproduktion]
Empfehlung	[Behebungsempfehlung]
Status	[Offen/In Bearbeitung/Geschlossen]
Behebungsdatum	[Datum oder N/A]

11.7. Vulnerability Disclosure Policy

11.7.1. Meldeverfahren

Sicherheitslücken können gemeldet werden an:

- E-Mail: security@worldiety.de
- PGP-Key: [Key-ID verfügbar auf Anfrage]

11.7.2. Reaktionszeiten

Schweregrad	Ziel-Reaktionszeit
Kritisch	24 Stunden
Hoch	72 Stunden
Mittel	7 Tage
Niedrig	30 Tage

11.7.3. Veröffentlichungspolitik

- Coordinated Disclosure nach 90 Tagen
- Benachrichtigung betroffener Kunden vor Veröffentlichung
- Security Advisory mit CVE-ID (falls zutreffend)

11.8. Security Notes Index

Die folgenden Security Notes sind im Quellcode dokumentiert:

Nr.	Datei:Zeile	Beschreibung
1	`flow/evt_repository_assigned.go:27`	Privilege Escalation Schutz
2	`session/ui/login.go:145`	User Enumeration Schutz bei Login
3	`management_mail.go:126`	User Enumeration Schutz bei Password-Reset
4	`management_mail.go:131`	Neue Security Codes bei Reset
5	`management_mail.go:169`	User Enumeration Schutz bei Verification
6	`management_mail.go:174`	Neue Verification Codes
7	`user/password.go:49`	Keine Sleep-Verzögerung wegen Argon2id
8	`user/password.go:52`	Trade-off bei schwachen Passwörtern
9	`dataimport/ui/page_select_parser.go:72`	Staging-ID nicht via Query
10	`usercircle/ui/view_users.go:32`	Circle-Admin UI-Schutz
11	`usercircle/ui/page_my_circle_groups.go:30`	Circle-Definition Berechtigung
12	`usercircle/ui/page_my_circle_roles.go:30`	Circle-Definition Berechtigung
13	`usercircle/ui/page_my_circle_licenses.go:30`	Circle-Definition Berechtigung
14	`usercircle/ui/page_my_circle_groups_users.go:26`	Circle-Definition Sichtbarkeit
15	`usercircle/ui/page_my_circle_roles_users.go:26`	Circle-Definition Sichtbarkeit
16	`usercircle/ui/page_my_circle_licenses_users.go:31`	Circle-Definition Sichtbarkeit
17	`usercircle/ui/page_my_circle_licenses_users.go:48`	Circle-Definition Zuweisung
18	`usercircle/ui/page_my_circle_licenses_users.go:56`	Circle-Definition Entzug
19	`drive/uc_read_drives.go:38`	Drive User-ID Zugriff
20	`secret/usecases.go:59`	Credentials Type-Safety

12. Incident Response Plan

12.1. Übersicht

Dieser Incident Response Plan beschreibt die Reaktionsverfahren für Sicherheitsvorfälle im Nago

Framework.

12.2. Eskalationspfad

```
@startuml
title Eskalationspfad bei Sicherheitsvorfällen

|L1 - Entwickler|
start
:Vorfall entdeckt;
:Erste Bewertung;
if (Schweregrad?) then (Kritisch/Hoch)
  |L2 - Security Lead|
  :Security Lead informieren;
  :Detailanalyse;
  if (Datenschutz betroffen?) then (Ja)
    |L3 - CISO/DSB|
    :CISO + Datenschutzbeauftragter;
    :Meldepflicht prüfen;
    if (Meldepflicht?) then (Ja)
      |L4 - Management/Legal|
      :Geschäftsführung + Rechtsabteilung;
      :Aufsichtsbehörde melden;
    endif
  endif
else (Mittel/Niedrig)
  :Ticket erstellen;
  :Im Sprint bearbeiten;
endif
stop

@enduml
```

12.3. Szenario 1: Master-Key-Kompromittierung

12.3.1. Beschreibung

Der 32-Byte Master Key für AES-GCM-256 Verschlüsselung wurde exponiert oder gestohlen.

12.3.2. Schweregrad

KRITISCH - Alle verschlüsselten Daten sind kompromittiert.

12.3.3. Detection

- Ungewöhnlicher Zugriff auf `.masterkey` Datei
- Umgebungsvariable `NAGO_MASTER_KEY` wurde exponiert

- Backup-Datei mit Master Key wurde geleakt
- Verdächtige Entschlüsselungsversuche im Log

12.3.4. Sofortmaßnahmen (< 1 Stunde)

Schritt	Aktion	Verantwortlich
1	**ALLE aktiven Sessions invalidieren** `session.Clear()` ausführen	Entwickler/Admin
2	**System offline nehmen** (falls notwendig)	Operations
3	**Neuen Master Key generieren** `backup.ReplaceMasterKey()` mit neuem 32-Byte Key	Security Lead
4	**Alten Master Key sperren**	Security Lead
5	**Forensische Kopie erstellen** Logs, Speicher, Netzwerkverkehr sichern	Security Lead

12.3.5. Containment (< 24 Stunden)

Schritt	Aktion	Verantwortlich
1	**Zugriffsanalyse** Wer hatte Zugriff auf den Key?	Security Lead
2	**Betroffene Daten identifizieren** Welche Stores waren verschlüsselt?	Entwickler
3	**Re-Encryption** Alle sensitiven Daten mit neuem Key verschlüsseln	Entwickler
4	**Passwort-Reset erzwingen** Alle Benutzer müssen Passwort ändern	Admin

12.3.6. Kommunikation

Zeitrahmen	Empfänger	Inhalt
< 1 Stunde	Internes Security-Team	Vollständige technische Details
< 24 Stunden	Geschäftsführung	Executive Summary, Impact Assessment
< 72 Stunden (DSGVO Art. 33)	Aufsichtsbehörde	Falls personenbezogene Daten betroffen
< 72 Stunden (DSGVO Art. 34)	Betroffene Nutzer	Falls hohes Risiko für Rechte und Freiheiten

12.3.7. Recovery

- Neuer Master Key in sicherer Umgebung generieren

- Key via HSM oder Secrets Manager speichern (empfohlen)
- Alle verschlüsselten Stores neu verschlüsseln
- System wieder online nehmen
- Monitoring intensivieren

12.3.8. Post-Incident

- Root Cause Analysis durchführen
- Dokumentation aktualisieren
- Lessons Learned Meeting
- Prozesse verbessern

12.4. Szenario 2: Backup-Datei-Leak

12.4.1. Beschreibung

Eine Backup-ZIP-Datei wurde unbefugt zugänglich.

12.4.2. Schweregrad

HOCH - Abhängig von enthaltenen Daten.

12.4.3. Detection

- Backup-Datei an unerwarteter Stelle gefunden
- Verdächtiger Download von Backup-Endpoint
- Externe Meldung über exponierte Daten

12.4.4. Sofortmaßnahmen (< 4 Stunden)

Schritt	Aktion	Verantwortlich
1	**Backup-Quelle identifizieren** Welches Backup wurde exponiert?	Security Lead
2	**Master Key Status prüfen** War der Master Key separat gespeichert?	Security Lead
3	**Risikobewertung** Verschlüsselte vs. unverschlüsselte Daten	Security Lead

12.4.5. Bewertungsmatrix

Situation	Risiko	Maßnahme
Master Key NICHT im Backup, separat gespeichert	MITTEL	Verschlüsselte Stores sind sicher, unverschlüsselte analysieren
Master Key WAR im Backup	KRITISCH	→ Szenario 1 aktivieren
Master Key unbekannt	HOCH	Annehmen, dass kompromittiert → Szenario 1

12.4.6. Kommunikation

- Intern: Security-Team sofort informieren
- Management: Innerhalb von 4 Stunden
- Datenschutzbeauftragter: Innerhalb von 24 Stunden
- Aufsichtsbehörde: Falls personenbezogene Daten unverschlüsselt betroffen (72h)

12.5. Szenario 3: Bootstrap-Admin-Missbrauch

12.5.1. Beschreibung

Der temporäre Bootstrap-Admin wurde für unbefugte Aktionen verwendet.

12.5.2. Schweregrad

KRITISCH - Administrator-Zugriff missbraucht.

12.5.3. Detection

- Bootstrap-Admin nach Ablaufzeit noch aktiv
- Unerwartete RBAC-Änderungen im Audit-Log
- Neue Administratoren ohne Genehmigung erstellt

12.5.4. Sofortmaßnahmen (< 30 Minuten)

Schritt	Aktion	Verantwortlich
1	**Bootstrap-Admin deaktivieren** Zeitlimit auf vergangenes Datum setzen	Admin
2	**Session des Angreifers terminieren** Alle Sessions des Bootstrap-Admin invalidieren	Admin

Schritt	Aktion	Verantwortlich
3	**Audit-Logs sichern** `evs` Event Store exportieren	Security Lead

12.5.5. Forensische Analyse

```
-- Pseudo-Query für Event Store Analyse
SELECT * FROM events
WHERE user_id = 'bootstrap-admin'
ORDER BY timestamp DESC;
```

Zu prüfen: * Welche Rollen wurden erstellt/geändert? * Welche Benutzer wurden erstellt? * Welche Permissions wurden zugewiesen? * Welche Daten wurden eingesehen/exportiert?

12.5.6. Rollback

- Alle vom Angreifer erstellten Benutzer deaktivieren
- Alle RBAC-Änderungen rückgängig machen
- Neue Bootstrap-Admin-Credentials generieren

12.6. Szenario 4: Session-Hijacking

12.6.1. Beschreibung

Ein Angreifer hat eine gültige Session übernommen.

12.6.2. Schweregrad

HOCH - Unbefugter Zugriff auf Benutzerkonto.

12.6.3. Detection

- Gleichzeitige Aktivität aus verschiedenen Locations
- Ungewöhnliches Benutzerverhalten
- Benutzer meldet nicht selbst durchgeführte Aktionen

12.6.4. Sofortmaßnahmen (< 15 Minuten)

Schritt	Aktion	Verantwortlich
1	**Betroffene Session terminieren** Session-ID aus Cookie invalidieren	Admin

Schritt	Aktion	Verantwortlich
2	**Benutzer informieren** Passwort-Reset empfehlen	Support
3	**TLS-Konfiguration prüfen** Secure/HttpOnly-Flags validieren	DevOps

12.6.5. Analyse

- Cookie-Domain-Konfiguration prüfen
- XSS-Schwachstellen suchen
- MITM-Möglichkeiten evaluieren
- Netzwerk-Logs analysieren

12.7. Szenario 5: Privilege Escalation via Repository-ID

12.7.1. Beschreibung

Ein Angreifer hat versucht, über manipulierte Repository-IDs Systemrechte zu erlangen.

12.7.2. Schweregrad

KRITISCH (bei Erfolg) / **NIEDRIG** (bei Blockierung)

12.7.3. Detection

- Validation Errors für `nago.*` Präfix im Log
- Ungewöhnliche Repository-Erstellungsversuche
- Security Note in `flow/evt_repository_assigned.go:27` wurde getriggert

12.7.4. Code-Referenz

```
// application/flow/evt_repository_assigned.go:27-29
if strings.HasPrefix(s, "nago.") {
    // security note: Schutz gegen Privilege Escalation
    return fmt.Errorf("repository id cannot start with 'nago.'")
}
```

12.7.5. Sofortmaßnahmen

Schritt	Aktion	Verantwortlich
1	**Angreifer identifizieren** Welcher Benutzer hat den Versuch unternommen?	Security Lead
2	**Benutzer sperren** Account deaktivieren, Session invalidieren	Admin
3	**Repository-Integrität prüfen** Keine System-Repositories manipuliert?	Entwickler

12.8. Kommunikationsvorlagen

12.8.1. Interne Benachrichtigung

```
BETREFF: [SECURITY INCIDENT] [SCHWEREGRAD] - Kurzbeschreibung

Zeitpunkt der Entdeckung: [DATUM UHRZEIT]
Schweregrad: [KRITISCH/HOCH/MITTEL/NIEDRIG]
Status: [AKTIV/CONTAINED/RESOLVED]

ZUSAMMENFASSUNG:
[Kurze Beschreibung des Vorfalls]

BETROFFENE SYSTEME:
- [System 1]
- [System 2]

SOFORTMA NAHMEN:
- [Maßnahme 1]
- [Maßnahme 2]

NÄCHSTE SCHRITTE:
- [Schritt 1]
- [Schritt 2]

KONTAKT:
Security Lead: [Name] - [Telefon]
```

12.8.2. Externe Benachrichtigung (Betroffene)

```
BETREFF: Wichtige Sicherheitsinformation zu Ihrem Konto

Sehr geehrte/r [NAME],

wir möchten Sie über einen Sicherheitsvorfall informieren, der
möglicherweise Ihre Daten betrifft.
```

```
WAS IST PASSIERT:
[Kurze, verständliche Beschreibung]

WELCHE DATEN SIND BETROFFEN:
[Liste der betroffenen Datentypen]

WAS WIR UNTERNOMMEN HABEN:
[Ergriffene Maßnahmen]

WAS SIE TUN SOLLTEN:
1. Ändern Sie Ihr Passwort unter [LINK]
2. Überprüfen Sie Ihre letzten Aktivitäten
3. [Weitere Empfehlungen]

KONTAKT:
Bei Fragen erreichen Sie uns unter:
- E-Mail: security@worldiety.de
- Telefon: [NUMMER]

Mit freundlichen Grüßen,
[Firma/Team]
```

12.8.3. Meldung an Aufsichtsbehörde (DSGVO Art. 33)

```
MELDUNG EINER DATENSCHUTZVERLETZUNG

Verantwortlicher: worldiety GmbH
Datenschutzbeauftragter: [Name, Kontakt]

Art der Verletzung: [Beschreibung]
Betroffene Datenkategorien: [Liste]
Ungefähre Anzahl betroffener Personen: [Zahl]
Ungefähre Anzahl betroffener Datensätze: [Zahl]

Wahrscheinliche Folgen: [Beschreibung]

Ergriffene Maßnahmen: [Liste]

Geplante Maßnahmen: [Liste]

Datum der Kenntniserlangung: [Datum]
```

12.9. Post-Incident Review Template

12.9.1. Zusammenfassung

Feld	Wert
Incident-ID	[INC-YYYY-NNN]
Zeitraum	[Start] - [Ende]
Schweregrad	[KRITISCH/HOCH/MITTEL/NIEDRIG]
Root Cause	[Kurzbeschreibung]
Impact	[Betroffene Systeme/Benutzer]

12.9.2. Timeline

Zeitpunkt	Ereignis
[T+0]	Vorfall entdeckt
[T+X]	Eskalation an Security Lead
[T+X]	Containment abgeschlossen
[T+X]	Recovery abgeschlossen

12.9.3. Lessons Learned

- Was hat gut funktioniert?
- Was hätte besser laufen können?
- Welche Prozesse müssen angepasst werden?
- Welche technischen Verbesserungen sind nötig?

12.9.4. Action Items

ID	Maßnahme	Verantwortlich	Deadline
1	[Maßnahme]	[Name]	[Datum]
2	[Maßnahme]	[Name]	[Datum]

12.10. Deployment-Security-Verweis

Dieser Incident Response Plan behandelt ausschließlich **Application Layer Security** des Nago Frameworks.

Für Deployment-bezogene Sicherheitsvorfälle (Infrastruktur, Netzwerk, Container) siehe die separate **Nago Hub Hosting Dokumentation**, die folgende Themen abdeckt:

- Reverse Proxy / Load Balancer Incidents
- TLS-Zertifikatsprobleme
- Firewall-Breach

- Container-Escape
- Host-System-Kompromittierung
- DDoS-Mitigation
- Disaster Recovery