Application Security Verification Standard

Version 5.0.0

May 2025

目次

[本標準について 9](#_Toc200894476)

[著作権とライセンス 9](#_Toc200894477)

[プロジェクトリーダー 9](#_Toc200894478)

[ワーキンググループ 9](#_Toc200894479)

[その他の主要執筆者 9](#_Toc200894480)

[その他の執筆者およびレビュー担当者 9](#_Toc200894481)

[序文 10](#_Toc200894482)

[はじめに 10](#_Toc200894483)

[バージョン 5.0 の背後にある原則 10](#_Toc200894484)

[今後の展望 10](#_Toc200894485)

[ASVS とは何か？ 11](#_Toc200894486)

[ASVS のスコープ 11](#_Toc200894487)

[アプリケーション (Application) 11](#_Toc200894488)

[セキュリティ (Security) 11](#_Toc200894489)

[検証 (Verification) 11](#_Toc200894490)

[標準 (Standard) 11](#_Toc200894491)

[要件 (Requirement) 12](#_Toc200894492)

[セキュリティ上の決定事項の文書化 12](#_Toc200894493)

[アプリケーションセキュリティ検証レベル 13](#_Toc200894494)

[レベル評価 13](#_Toc200894495)

[レベル 1 13](#_Toc200894496)

[レベル 2 13](#_Toc200894497)

[レベル 3 14](#_Toc200894498)

[どのレベルを達成するか 14](#_Toc200894499)

[ASVS の使い方 14](#_Toc200894500)

[ASVS の構成 14](#_Toc200894501)

[リリース戦略 14](#_Toc200894502)

[ASVS の柔軟性 14](#_Toc200894503)

[ASVS 要件の見方 15](#_Toc200894504)

[ASVS のフォーク 15](#_Toc200894505)

[ASVS のユースケース 15](#_Toc200894506)

[詳細なセキュリティアーキテクチャガイダンスとして 15](#_Toc200894507)

[専用のセキュアコーディングリファレンスとして 16](#_Toc200894508)

[自動ユニットテストおよび自動統合テストのガイドとして 16](#_Toc200894509)

[セキュア開発トレーニングのために 16](#_Toc200894510)

[セキュアなソフトウェアの調達をガイドするためのフレームワークとして 16](#_Toc200894511)

[実際に ASVS を適用する 16](#_Toc200894512)

[監査と認証 17](#_Toc200894513)

[ASVS 認証と認証マークに対する OWASP の見解 17](#_Toc200894514)

[ASVS コンプライアンスの検証方法 17](#_Toc200894515)

[検証レポート 17](#_Toc200894516)

[検証のスコープ 17](#_Toc200894517)

[検証メカニズム 17](#_Toc200894518)

[v4.x と比較した変更点 19](#_Toc200894519)

[はじめに 19](#_Toc200894520)

[要件の理念 19](#_Toc200894521)

[スコープとフォーカス 19](#_Toc200894522)

[メカニズムよりもセキュリティ目標を重視 19](#_Toc200894523)

[セキュリティ上の決定事項の文書化 19](#_Toc200894524)

[構造変更と新章 19](#_Toc200894525)

[他の標準への直接マッピングの削除 20](#_Toc200894526)

[NIST Digital Identity Guidelines との結びつきの低減 20](#_Toc200894527)

[共通脆弱性タイプ一覧 (Common Weakness Enumeration, CWE) からの脱却 20](#_Toc200894528)

[レベル定義の再考 20](#_Toc200894529)

[より容易なエントリレベル 21](#_Toc200894530)

[試験性の誤り 21](#_Toc200894531)

[リスクだけではない 21](#_Toc200894532)

[V1 エンコーディングとサニタイゼーション 22](#_Toc200894533)

[管理目標 22](#_Toc200894534)

[V1.1 エンコーディングおよびサニタイゼーションアーキテクチャ 22](#_Toc200894535)

[V1.2 インジェクション防御 22](#_Toc200894536)

[V1.3 サニタイゼーション 23](#_Toc200894537)

[V1.4 メモリ、文字列、アンマネージドコード 25](#_Toc200894538)

[V1.5 安全なデシリアライゼーション 25](#_Toc200894539)

[参考情報 26](#_Toc200894540)

[V2 バリデーションとビジネスロジック 27](#_Toc200894541)

[管理目標 27](#_Toc200894542)

[V2.1 バリデーションとビジネスロジックドキュメント 27](#_Toc200894543)

[V2.2 入力バリデーション 27](#_Toc200894544)

[V2.3 ビジネスロジックのセキュリティ 28](#_Toc200894545)

[V2.4 アンチオートメーション 29](#_Toc200894546)

[参考情報 29](#_Toc200894547)

[V3 Web フロントエンドセキュリティ 30](#_Toc200894548)

[管理目標 30](#_Toc200894549)

[V3.1 Web フロントエンドセキュリティドキュメント 30](#_Toc200894550)

[V3.2 意図しないコンテンツ解釈 30](#_Toc200894551)

[V3.3 クッキーセットアップ 30](#_Toc200894552)

[V3.4 ブラウザのセキュリティメカニズムヘッダ 31](#_Toc200894553)

[V3.5 ブラウザのオリジン分離 32](#_Toc200894554)

[V3.6 外部リソース完全性 33](#_Toc200894555)

[V3.7 ブラウザのセキュリティに関するその他の考慮事項 34](#_Toc200894556)

[参考情報 34](#_Toc200894557)

[V4 API と Web サービス 35](#_Toc200894558)

[管理目標 35](#_Toc200894559)

[V4.1 一般的な Web サービスセキュリティ 35](#_Toc200894560)

[V4.2 HTTP メッセージ構造バリデーション 35](#_Toc200894561)

[V4.3 GraphQL 36](#_Toc200894562)

[V4.4 WebSocket 36](#_Toc200894563)

[参考情報 37](#_Toc200894564)

[V5 ファイル処理 38](#_Toc200894565)

[管理目標 38](#_Toc200894566)

[V5.1 ファイル処理ドキュメント 38](#_Toc200894567)

[V5.2 ファイルアップロードとコンテンツ 38](#_Toc200894568)

[V5.3 ファイル保存 39](#_Toc200894569)

[V5.4 ファイルダウンロード 39](#_Toc200894570)

[参考情報 39](#_Toc200894571)

[V6 認証 41](#_Toc200894572)

[管理目標 41](#_Toc200894573)

[V6.1 認証ドキュメント 41](#_Toc200894574)

[V6.2 パスワードセキュリティ 41](#_Toc200894575)

[V6.3 一般的な認証セキュリティ 42](#_Toc200894576)

[V6.4 認証要素のライフサイクルとリカバリ 43](#_Toc200894577)

[V6.5 一般的な多要素認証要件 44](#_Toc200894578)

[V6.6 経路外認証メカニズム 45](#_Toc200894579)

[V6.7 暗号認証メカニズム 46](#_Toc200894580)

[V6.8 アイデンティティプロバイダによる認証 46](#_Toc200894581)

[参考情報 47](#_Toc200894582)

[V7 セッション管理 48](#_Toc200894583)

[管理目標 48](#_Toc200894584)

[V7.1 セッション管理ドキュメント 48](#_Toc200894585)

[V7.2 基本セッション管理セキュリティ 49](#_Toc200894586)

[V7.3 セッションタイムアウト 49](#_Toc200894587)

[V7.4 セッションの終了 49](#_Toc200894588)

[V7.5 セッションの悪用に対する防御 50](#_Toc200894589)

[V7.6 フェデレーション再認証 50](#_Toc200894590)

[参考情報 51](#_Toc200894591)

[V8 認可 52](#_Toc200894592)

[管理目標 52](#_Toc200894593)

[V8.1 認可ドキュメント 52](#_Toc200894594)

[V8.2 一般的な認可設計 52](#_Toc200894595)

[V8.3 操作レベルの認可 53](#_Toc200894596)

[V8.4 他の認可の考慮 53](#_Toc200894597)

[参考情報 54](#_Toc200894598)

[V9 自己完結型トークン 55](#_Toc200894599)

[管理目標 55](#_Toc200894600)

[V9.1 トークンのソースと完全性 55](#_Toc200894601)

[V9.2 トークンコンテンツ 55](#_Toc200894602)

[参考情報 56](#_Toc200894603)

[V10 OAuth と OIDC 57](#_Toc200894604)

[管理目標 57](#_Toc200894605)

[V10.1 一般的な OAuth と OIDC セキュリティ 58](#_Toc200894606)

[V10.2 OAuth クライアント 58](#_Toc200894607)

[V10.3 OAuth リソースサーバ 59](#_Toc200894608)

[V10.4 OAuth 認可サーバ 60](#_Toc200894609)

[V10.5 OIDC クライアント 61](#_Toc200894610)

[V10.6 OpenID プロバイダ 62](#_Toc200894611)

[V10.7 同意管理 62](#_Toc200894612)

[参考情報 63](#_Toc200894613)

[V11 暗号化 64](#_Toc200894614)

[管理目標 64](#_Toc200894615)

[V11.1 暗号インベントリとドキュメント 64](#_Toc200894616)

[V11.2 安全な暗号の実装 65](#_Toc200894617)

[V11.3 暗号アルゴリズム 65](#_Toc200894618)

[V11.4 ハッシュ化とハッシュベース関数 66](#_Toc200894619)

[V11.5 乱数値 67](#_Toc200894620)

[V11.6 公開鍵暗号 67](#_Toc200894621)

[V11.7 使用中のデータの暗号化 67](#_Toc200894622)

[参考情報 68](#_Toc200894623)

[V12 安全な通信 69](#_Toc200894624)

[管理目標 69](#_Toc200894625)

[V12.1 一般的な TLS セキュリティガイダンス 69](#_Toc200894626)

[V12.2 外部向けサービスとの HTTPS 通信 69](#_Toc200894627)

[V12.3 一般的なサービス間通信セキュリティ 70](#_Toc200894628)

[参考情報 70](#_Toc200894629)

[V13 構成 71](#_Toc200894630)

[管理目標 71](#_Toc200894631)

[V13.1 構成ドキュメント 71](#_Toc200894632)

[V13.2 バックエンド通信構成 71](#_Toc200894633)

[V13.3 シークレット管理 72](#_Toc200894634)

[V13.4 意図しない情報漏洩 73](#_Toc200894635)

[参考情報 73](#_Toc200894636)

[V14 データ保護 74](#_Toc200894637)

[管理目標 74](#_Toc200894638)

[V14.1 データ保護ドキュメント 74](#_Toc200894639)

[V14.2 一般的なデータ保護 74](#_Toc200894640)

[V14.3 クライアントサイドのデータ保護 75](#_Toc200894641)

[参考情報 76](#_Toc200894642)

[V15 セキュアコーディングとアーキテクチャ 77](#_Toc200894643)

[管理目標 77](#_Toc200894644)

[V15.1 セキュアコーディングとアーキテクチャドキュメント 77](#_Toc200894645)

[V15.2 セキュリティアーキテクチャと依存関係 78](#_Toc200894646)

[V15.3 防御的コーディング 78](#_Toc200894647)

[V15.4 安全な同時並行性 79](#_Toc200894648)

[参考情報 80](#_Toc200894649)

[V16 セキュリティログ記録とエラー処理 81](#_Toc200894650)

[管理目標 81](#_Toc200894651)

[V16.1 セキュリティログ記録ドキュメント 81](#_Toc200894652)

[V16.2 一般的なログ記録 81](#_Toc200894653)

[V16.3 セキュリティイベント 82](#_Toc200894654)

[V16.4 ログ保護 82](#_Toc200894655)

[V16.5 エラー処理 83](#_Toc200894656)

[参考情報 83](#_Toc200894657)

[V17 WebRTC 85](#_Toc200894658)

[管理目標 85](#_Toc200894659)

[V17.1 TURN サーバ 85](#_Toc200894660)

[V17.2 メディア 86](#_Toc200894661)

[V17.3 シグナリング 87](#_Toc200894662)

[参考情報 87](#_Toc200894663)

[付録 A: 用語集 88](#_Toc200894664)

[付録 B: 参考情報 94](#_Toc200894665)

[OWASP 主要プロジェクト 94](#_Toc200894666)

[OWASP チートシートシリーズプロジェクト 94](#_Toc200894667)

[モバイルセキュリティ関連プロジェクト 94](#_Toc200894668)

[OWASP Internet of Things 関連プロジェクト 94](#_Toc200894669)

[OWASP Serverless プロジェクト 94](#_Toc200894670)

[その他 94](#_Toc200894671)

[付録 C: 暗号化標準 95](#_Toc200894672)

[暗号インベントリとドキュメント 95](#_Toc200894673)

[暗号パラメータの等価な強度 95](#_Toc200894674)

[乱数値 96](#_Toc200894675)

[暗号アルゴリズム 96](#_Toc200894676)

[AES 暗号モード 97](#_Toc200894677)

[鍵ラッピング 98](#_Toc200894678)

[認証された暗号 98](#_Toc200894679)

[ハッシュ関数 99](#_Toc200894680)

[一般的なユースケースでのハッシュ関数 99](#_Toc200894681)

[パスワード保存のためのハッシュ関数 100](#_Toc200894682)

[鍵導出関数 (KDF) 100](#_Toc200894683)

[汎用の鍵導出関数 100](#_Toc200894684)

[パスワードベースの鍵導出関数 100](#_Toc200894685)

[鍵交換メカニズム 101](#_Toc200894686)

[KEX スキーム 101](#_Toc200894687)

[Diffie-Hellman グループ 101](#_Toc200894688)

[メッセージ認証コード (MAC) 102](#_Toc200894689)

[デジタル署名 103](#_Toc200894690)

[ポスト量子暗号標準 103](#_Toc200894691)

[付録 D: 推奨事項 104](#_Toc200894692)

[はじめに 104](#_Toc200894693)

[推奨される、スコープ内のメカニズム 104](#_Toc200894694)

[ソフトウェアセキュリティの原則 104](#_Toc200894695)

[ソフトウェアセキュリティプロセス 105](#_Toc200894696)

[付録 E - 貢献者 106](#_Toc200894697)

口絵

## 本標準について

アプリケーションセキュリティ検証標準はアーキテクト、開発者、テスト担当者、セキュリティ専門家、ツールベンダ、利用者が安全なアプリケーションの定義、構築、テスト、検証に使用できるアプリケーションセキュリティ要件のリストです。

## 著作権とライセンス

Version 5.0.0, May 2025



Copyright © 2008-2025 The OWASP Foundation.

本書は [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/) に基づいてリリースされています。

再利用または配布する場合には、本著作物のライセンス条項を他者に対して明確に伝える必要があります。

## プロジェクトリーダー

|  |  |
| --- | --- |
| Elar Lang | Josh C Grossman |
| Jim Manico | Daniel Cuthbert |

## ワーキンググループ

|  |  |  |  |
| --- | --- | --- | --- |
| Tobias Ahnoff | Ralph Andalis | Ryan Armstrong | Gabriel Corona |
| Meghan Jacquot | Shanni Prutchi | Iman Sharafaldin | Eden Yardeni |

## その他の主要執筆者

|  |  |
| --- | --- |
| Sjoerd Langkemper | Isaac Lewis |
| Mark Carney | Sandro Gauci |

## その他の執筆者およびレビュー担当者

付録 E にその他の執筆者のリストがあります。

クレジットが 5.x クレジットリストにない場合は、将来の 5.x アップデートで認識されるように GitHub のチケットを記録してください。

アプリケーションセキュリティ検証標準は ASVS 1.0 (2008) から 4.0 (2019) に至る関係者の成果に基づいて構築しています。現在も ASVS に残っている構成と検証項目の多くはもともと Andrew van der Stock, Mike Boberski, Jeff Williams, Dave Wichers をはじめとする多くの執筆者によって書かれました。これまで貢献してくださったすべての皆様に感謝します。以前の執筆者の包括的なリストについては、以前の各バージョンを確認してください。

# 序文

アプリケーションセキュリティ検証標準 (ASVS) バージョン 5.0 へようこそ。

## はじめに

ASVS は、もともと 2008 年にコミュニティの共同作業を通じて開始され、現代の Web アプリケーションとサービスを設計、開発、テストするための包括的な一連のセキュリティ要件を定義しています。

2019 年の ASVS 4.0 のリリースと 2021 年のマイナーアップデート (v4.0.3) に続き、バージョン 5.0 はソフトウェアセキュリティの最新の進歩を反映して近代化された重要なマイルストーンとなります。

ASVS 5.0 は、この重要な標準を更新および改善するために、プロジェクトリーダー、ワーキンググループメンバー、およびより広範な OWASP コミュニティからの多大な貢献の結果です。

## バージョン 5.0 の背後にある原則

このメジャーリビジョンはいくつかの重要な原則を念頭に置いて策定されました。

* スコープと焦点の洗練: このバージョンの標準は、その名称にある、アプリケーション (Application)、セキュリティ (Security)、検証 (Verification)、標準 (Standard) という基本的な柱に、より直接的に沿うように設計されています。要件は、特定の技術的実装を義務付けるのではなく、セキュリティ上の欠陥の防止に重点を置くように書き直されました。要件文はその存在理由を示す自明なものになるよう目指しています。
* セキュリティに関する決定事項の文書化のサポート: ASVS 5.0 では主要なセキュリティに関する決定事項を文書化するための要件を導入しています。これによりトレーサビリティを向上し、状況に応じた実装をサポートするため、組織はセキュリティ態勢を特定のニーズやリスクに合わせて調整できます。
* レベルの更新: ASVS は三層モデルを維持していますが、レベル定義は ASVS を採用しやすくなるように進化しています。レベル 1 は ASVS 採用の最初のステップとして設計されており、防御の第一層を提供します。レベル 2 は標準的なセキュリティプラクティスの包括的な視点を表し、レベル 3 は高度な高保証要件に対応しています。
* コンテンツの再構成と拡張: ASVS 5.0 は 17 の章にわたって約 350 の要件を含みます。章はわかりやすさと使いやすさのために再編成されました。v4.0 と v5.0 の間の双方向マッピングが提供され、移行が容易になります。

## 今後の展望

アプリケーションのセキュリティ確保が決して完全に完了しないのと同様に、ASVS もまた決して完了することはありません。バージョン 5.0 はメジャーリリースですが、開発は継続しています。このリリースは、より広範なコミュニティがこれまで蓄積されてきた改善点や追加点から恩恵を受けることができるだけでなく、将来の強化のための基盤も築いています。これにはコア要件セットの上に構築された実装および検証ガイダンスを作成するためのコミュニティ主導の取り組みを含む可能性があります。

ASVS 5.0 は安全なソフトウェア開発のための信頼できる基盤となるように設計されています。アプリケーションセキュリティの現状を共に前進させるために、コミュニティがこの標準を採用、貢献、構築するようお誘いしています。

# ASVS とは何か？

アプリケーションセキュリティ検証標準 (ASVS) はウェブアプリケーションとサービスのセキュリティ要件を定義しており、安全なアプリケーションを設計、開発、保守、またはそれらのセキュリティを評価することを目的とするすべての人にとって貴重なリソースとなります。

この章では ASVS のスコープ、優先度ベースのレベルの構造、標準の主な使用例など、ASVS を使用する上での不可欠な側面について概説します。

## ASVS のスコープ

ASVS のスコープは、アプリケーション (Application)、セキュリティ (Security)、検証 (Verification)、標準 (Standard) という名称で定義されます。これは、達成する必要があるセキュリティ原則を特定するという包括的な目標を掲げ、どの要件を含めるか、または除外するかを定めます。このスコープでは、実装要件の基盤となるドキュメント要件も考慮します。

攻撃者にとってスコープというようなものは存在しません。したがって、ASVS 要件は、CI/CD プロセス、ホスティング、運用アクティビティなど、アプリケーションライフサイクルの他の側面に関するガイダンスと併せて評価する必要があります。

### アプリケーション (Application)

ASVS では「アプリケーション」を、セキュリティコントロールを統合する必要がある、開発中のソフトウェア製品と定義しています。ASVS は、開発ライフサイクルのアクティビティを規定したり、CI/CD パイプラインを介してアプリケーションを構築する方法を指示するのではなく、製品自体の中で達成する必要があるセキュリティ成果を指定します。

Web アプリケーションファイアウォール (WAF)、ロードバランサ、プロキシなど、HTTP トラフィックを提供、変更、検証するコンポーネントは、一部のセキュリティ制御がこれらに直接依存しているか、これらを介して実装できるため、特定の目的のためにアプリケーションの一部とみなされるかもしれません。これらのコンポーネントは、キャッシュされたレスポンス、レート制限、あるいは送信元と送信先に基づく送受信接続の制限に関連する要件について考慮する必要があります。

逆に、ASVS では通常、アプリケーションに直接関係しない要件や、構成がアプリケーションの責任範囲外となる要件を除外します。たとえば、DNS の問題は一般的に別のチームや機能によって管理されます。

同様に、アプリケーションが入力をどのように消費し、出力をどのように生成するかについて責任を負いますが、外部プロセスがアプリケーションやそのデータとやり取りする場合、ASVS のスコープ外とみなされます。たとえば、アプリケーションやそのデータのバックアップは、通常、外部プロセスの責任であり、アプリケーションやその開発者によって制御されるものではありません。

### セキュリティ (Security)

すべての要件はセキュリティに明白な影響を与える必要があります。要件がない場合、アプリケーションは安全でなくなり、要件を実装することで、セキュリティリスクの発生可能性または影響のいずれかを軽減する必要があります。

機能的側面、コードスタイル、ポリシー要件など、その他の考慮事項はスコープ外です。

### 検証 (Verification)

要件は検証可能である必要があり、検証の結果は「不合格 (fail)」または「合格 (pass)」の判定になる必要があります。

### 標準 (Standard)

ASVS は標準に準拠するために実装されるセキュリティ要件の集合として設計されています。つまり、要件はそれを達成するためのセキュリティ目標を定義することに限定されます。その他の関連情報は ASVS 上に構築することも、マッピングを介してリンクすることもあります。

具体的には、OWASP には多くのプロジェクトがあり、ASVS は他のプロジェクトの内容との重複を意図的に避けています。たとえば、開発者は「特定の技術や環境で特定の要件を実装するにはどのようにすればよいか」という疑問を持つかもしれません。これは Cheat Sheet Series プロジェクトでカバーすべきです。検証者は「この環境でこの要件をテストするにはどのようにすればよいか」という疑問を持つかもしれません。これは Web Security Testing Guide プロジェクトでカバーすべきです。

ASVS はセキュリティ専門家だけが使用することを意図しているわけではありませんが、読者には内容を理解するための技術的知識や特定の概念を調査する能力が求められます。

### 要件 (Requirement)

特に ASVS では要件 (requirement) という用語が使用されており、これを満たすために何を達成する必要があるかを説明しています。ASVS には主な条件として要件 (must) のみを含み、推奨事項 (should) は含みません。

言い換えると、推奨事項は、それが問題を解決するための多くの可能な選択肢の一つにすぎない場合や、コードスタイルの考慮事項である場合など、要件である定義を満たしません。

ASVS の要件は、実装や技術に特化しすぎることなく、特定のセキュリティ原則に対応すること、と同時に、その存在理由が自明であることも意図しています。これはまた、要件が特定の検証方法や実装に基づいて構築されていないことも意味しています。

### セキュリティ上の決定事項の文書化

ソフトウェアセキュリティでは、セキュリティ設計と使用するメカニズムを早い段階で計画することで、完成した製品や機能においてより一貫性と信頼性の高い実装につながります。

さらに、特定の要件については、実装が複雑になり、アプリケーションのニーズに非常に特化したものになります。よくある例としてはパーミッション、入力バリデーション、さまざまなレベルの機密データに関する保護コントロールなどがあります。

これを考慮して、「すべてのデータを暗号化する必要がある」というような包括的な記述や、要件であらゆるユースケースをカバーしようとするのではなく、アプリケーション開発者によるこの種のコントロールへのアプローチと構成を文書化することを義務付けるドキュメント要件を盛り込みました。そして、これが適切かどうかをレビューし、実際の実装をドキュメントと比較して、実装が期待通りかどうかを評価できます。

これらの要件は、アプリケーションを開発する組織が特定のセキュリティ要件を実装する方法に関する決定事項を文書化することを意図しています。

ドキュメント要件は、常に章の最初のセクションにあります (すべての章にあるわけではありません)。文書化された決定事項を実際に実施する必要がある箇所には、必ず関連する実装要件があります。ここでのポイントは、ドキュメントが整備されていることと、実際の実装が二つの別のアクティビティであることを検証することです。

これらの要件を盛り込む主な要因は二つあります。一つ目の要因は、セキュリティ要件には、アップロードできるファイルの種類、適用すべきビジネスコントロール、特定フィールドに許可される文字など、ルールが適用されることがよくあることです。これらのルールはアプリケーションごとに異なるため、ASVS では規範的に定義することはできず、チートシートやより詳しい回答もこの場合には役に立ちません。同様に、これらの決定事項が文書化されていなければ、これらの決定事項を実装する要件の検証を行うことはできないでしょう。

二つ目の要因は、特定の要件において、アプリケーション開発では特定のセキュリティ課題への対処方法に関する柔軟性を提供することが重要となることです。たとえば、以前の ASVS バージョンでは、セッションタイムアウトのルールは非常に規範的でした。現実的に言えば、多くのアプリケーション、特にコンシューマ向けのものでは、より穏やかなルールを持ち、代わりに他の緩和策を実装することを好みます。したがって、ドキュメント要件はこのあたりの柔軟性を明示的に許容します。

明らかに、個々の開発者がこれらの決定を下して文書化することは期待されていません。むしろ、組織全体がそのような決定を下し、それを開発者に確実に伝えて、開発者がそれに従うようにします。

開発者に新機能の仕様と設計を提供することは、ソフトウェア開発の標準的な手順です。同様に、開発者はその都度独自の判断を下すのではなく、共通のコンポーネントとユーザインタフェースメカニズムを使用することが期待されています。そのため、これをセキュリティに拡張することは、驚くべきことではなく、議論の余地もないでしょう。

これを実現する方法にも柔軟性があります。セキュリティに関する決定事項は、リテラルドキュメントに文書化され、開発者が参照することが求められるかもしれません。あるいは、セキュリティに関する決定事項を文書化し、共通のコードライブラリに実装して、すべての開発者が使用することを義務付けることもあります。いずれの場合でも、望ましい結果が得られます。

## アプリケーションセキュリティ検証レベル

ASVS では三つのセキュリティ検証レベルを定義しており、レベルごとに深さと複雑さを増していきます。一般的な目標は、組織が最も重大なセキュリティ上の懸念に対処するために最初のレベルから始め、それから組織とアプリケーションのニーズに応じて上位レベルへ移行することです。レベルは文書および要件テキストにおいて L1, L2, L3 と表記することがあります。

各 ASVS レベルは、そのレベルで達成するために必要なセキュリティ要件を示し、残りの上位レベルの要件は推奨事項となります。

要件の重複や、上位レベルでは関係のない要件を避けるため、一部の要件は特定のレベルに適用されますが、上位レベルではより厳しい条件となっています。

### レベル評価

レベルは、セキュリティ要件の実装とテストの経験に基づき、各要件の優先度ベースの評価によって定義されます。主な焦点はリスク軽減と要件実装の労力を比較することです。もう一つの重要な要素は参入障壁を低く抑えることです。

リスク軽減は、その要件がアプリケーション内のセキュリティリスクのレベルをどの程度軽減するかを考慮します。その際、従来の機密性、完全性、可用性の影響因子を考慮するとともに、その要件が防御の主要な層であるか、多層防御とみなされるかを考慮します。

基準とレベル分けの決定に関する厳密な議論の結果、あらゆる状況に 100% 適合するわけではないことを受け入れつつ、大半のケースに当てはまる割り当てを導き出しました。つまり、場合によっては、組織は独自のリスク考慮事項に基づいて、早い段階で上位レベルの要件を優先付けることが望ましいこともあります。

各レベルにおける要件の種類は以下のように特徴付けることができます。

### レベル 1

このレベルはアプリケーションを保護にする際に考慮すべき最低限の要件を含み、重要な出発点となります。このレベルには ASVS 要件の約 20% を含みます。このレベルの目標は、可能な限り要件を少なくし、参入障壁を低く抑えることです。

これらの要件は通常、悪用可能な他の脆弱性や前提条件を必要としない一般的な攻撃を防ぐための、重要または基本的な第一層の防御要件です。

第一層の防御要件に加えて、パスワードに関連する要件など、上位レベルでは影響が少ない要件もあります。上位レベルでは多要素認証要件が重要になるため、これらはレベル 1 ではより重要です。

レベル 1 は、要件の数が少ないため検証は容易になるものの、ドキュメントやコードへの内部アクセスを持たない外部のテスト担当者によるペネトレーションテスト (「ブラックボックス」テストなど) が必ずしも可能であるとは限りません。

### レベル 2

ほとんどはアプリケーションはこのレベルのセキュリティを達成するように努力すべきです。ASVS の要件の約 50% は L2 であるため、アプリケーションが L2 に準拠するには ASVS の要件の約 70% (L1 と L2 要件のすべて) を実装する必要があります。

これらの要件は通常、あまり一般的ではない攻撃、または一般的な攻撃に対するより複雑な防御のいずれかに関連します。これらは依然として防御の第一層であるかもしれませんし、攻撃が成功するには特定の前提条件を必要とするかもしれません。

### レベル 3

このレベルは、最高レベルのセキュリティを実証しようとするアプリケーションの目標とすべきであり、準拠すべき要件の最後の約 30% を提供します。

このセクションの要件は通常、多層防御メカニズムまたはその他の有用だが実装が困難なコントロールのいずれかです。

### どのレベルを達成するか

優先度ベースのレベルは、組織とアプリケーションのアプリケーションセキュリティ成熟度を反映することを意図しています。アプリケーションがどのレベルに到達すべきかを ASVS が規定するのではなく、組織はそのリスクを分析し、アプリケーションの機密性や、もちろんアプリケーションのユーザの期待に応じて、どのレベルに到達すべきかを決定する必要があります。

たとえば、限定的な機密データしか収集していないアーリーステージのスタートアップ企業は初期のセキュリティ目標としてレベル 1 に絞ることに決めるかもしれませんが、銀行はオンラインバンキングアプリケーションでレベル 3 未満のものを顧客に正当化するのは難しいかもしれません。

## ASVS の使い方

### ASVS の構成

ASVS は合計約 350 の要件で構成されており、17 の章に分かれています。各賞はさらにセクションに分かれています。

章とセクションを分ける目的は、アプリケーションに関連するものに基づいて、章とセクションの選択や除外を簡単にすることです。たとえば、マシン間 API の場合、Web フロントエンドに関連する V3 章の要件は関係ありません。OAuth や WebRTC を使用しないのであれば、それらの章も無視できます。

### リリース戦略

ASVS リリースは "Major.Minor.Patch" というパターンに従い、番号はリリース内での変更内容に関する情報を示します。メジャーリリースでは最初の番号が変わり、マイナーリリースでは二番目の番号が変わり、パッチリリースでは三番目の番号が変わります。

* メジャーリリース - 全面的な再編成。要件番号を含め、ほぼすべてが変わっている可能性があります。準拠のための再評価が必要になります (例: 4.0.3 -> 5.0.0)。
* マイナーリリース - 要件が追加または削除されている可能性がありますが、全体的な番号付けは変わりません。準拠のための再評価が必要になりますが、より簡単になるはずです (例: 5.0.0 -> 5.1.0)。
* パッチリリース - 要件が削除されている (たとえば、重複していたり陳腐化した場合) または厳しさが緩和されている可能性がありますが、以前のリリースで準拠していたアプリケーションはパッチリリースでも準拠します (例: 5.0.0 -> 5.0.1)。

上記は特に ASVS の要件に関連しています。周囲のテキストや付録などのその他のコンテンツへの変更は、互換性を損なう変更とはみなされません。

### ASVS の柔軟性

ドキュメント要件やレベルメカニズムなど、上述したいくつかの点により、ASVS をより柔軟かつ組織固有の方法で使用できるようになります。

さらに、組織には、アプリケーションの特定の特性とリスクレベルに基づいて要件を調整するために、組織またはドメイン固有のフォークを作成することを強く推奨します。ただし、要件 4.1.1 に合格することが、すべてのバージョンで同じ意味になるように、トレーサビリティを維持することが重要です。

理想的には、各組織が独自に調整した ASVS を作成して、関係のないセクション (GraphQL, WebSockets, SOAP など、不要な場合) を省略すべきです。組織固有の ASVS バージョンまたはサプリメントは、組織固有の実装ガイダンスを提供し、要件に準拠する際に使用するライブラリやリソースを詳述するのにも適しています。

### ASVS 要件の見方

各要件には <chapter>.<section>.<requirement> という形式の識別子があります。各要素は数値です。例: 1.11.3

* <chapter> の値は要件の属する章に対応します。例: 1.#.# の要件はすべて エンコーディングとサニタイゼーション の章のものです。
* <section> の値は要件が現れる章内のセクションに対応します。例: 1.2.# の要件はすべて エンコーディングとサニタイゼーション の章の インジェクション防御 セクションにあります。
* <requirement> の値は章およびセクション内の特定の要件を識別します。例: この標準のバージョン 5.0.0 での 1.2.5 は以下のとおりです。

アプリケーションが OS コマンドインジェクションに対して保護していること、およびオペレーティングシステムコールがパラメータ化された OS クエリを使用するか、コンテキストに応じたコマンドライン出力エンコーディングを使用する。

識別子は標準のバージョン間で変更となる可能性があるため、他のドキュメント、レポート、ツールでは v<version>-<chapter>.<section>.<requirement> という形式を使用することを推奨します。ここでは 'version' は ASVS バージョンタグです。例: v5.0.0-1.2.5 はバージョン 5.0.0 の 'エンコーディングとサニタイゼーション' の章の 'インジェクション防御' セクションの 5 番目にある要件を意味すると理解できます。 (これは v<version>-<requirement\_identifier> と要約できます。)

注: この形式でバージョン番号の前にある v は常に小文字にする必要があります。

v<version> 要素を含めずに識別子を使用する場合には、最新のアプリケーションセキュリティ検証標準コンテンツを参照していると想定すべきです。標準の進化や変更に伴いこれが問題になるため、執筆者や開発者はバージョン要素を含める必要があります。

ASVS 要件リストは CSV、JSON、および参照またはプログラムでの使用に役立つ可能性があるその他の形式で提供されています。

### ASVS のフォーク

組織は ASVS を採用することでメリットがあります。三つのレベルのいずれかを選択するか、ドメイン固有のフォークを作成して、アプリケーションリスクレベルごとに要件を調整します。トレーサビリティが維持されている限りこのようなフォークを推奨します。要件 4.1.1 に合格することがすべてのバージョンで同じことを意味するようにします。

理想的には、各組織が独自に調整した ASVS を作成し、関係のないセクション (未使用の場合、GraphQL、Websockets、SOAP など) を省略すべきです。フォークは ASVS レベル 1 をベースラインとして始めて、アプリケーションのリスクに基づいてレベル 2 または 3 に進めるべきです。

## ASVS のユースケース

ASVS はアプリケーションのセキュリティを評価するために使用できます。これについては次の章で詳しく説明しますが、ASVS (もしくはフォーク版) のその他の潜在的な用途がいくつか考えられています。

### 詳細なセキュリティアーキテクチャガイダンスとして

アプリケーションセキュリティ検証標準のより一般的な用途の一つはセキュリティアーキテクトのためのリソースです。特に最新のアプリケーションでは、安全なアプリケーションアーキテクチャの構築方法について利用できるリソースが限られています。ASVS を使用して、セキュリティアーキテクトがデータ保護パターンや入力バリデーション戦略などの一般的な問題に対してより適切なコントロールを選択できるようにすることで、これらのギャップを埋めることができます。アーキテクチャとドキュメントの要件は、特にこのために役立ちます。

### 専用のセキュアコーディングリファレンスとして

ASVS はアプリケーション開発時にセキュアコーディングリファレンスを作成するための基盤として使用でき、開発者がソフトウェアを構築する際にセキュリティを考慮するよう支援します。ASVS をベースとすることも可能ですが、組織は明確で統一された独自のガイダンスを作成すべきです。理想的にはセキュリティエンジニアやセキュリティアーキテクトの指導に基づいて作成します。この延長として、組織は可能な限り、ガイダンスの中で参照でき、開発者が使用できる承認済みセキュリティメカニズムとライブラリを準備することが推奨されます。

### 自動ユニットテストおよび自動統合テストのガイドとして

ASVS は高度にテスト可能なように設計されています。検証の一部は技術的なものですが、他の要件 (アーキテクチャ要件やドキュメント要件など) ではドキュメントやアーキテクチャのレビューが必要になることがあります。技術的な手段で検証可能な要件に関連する、特定の関連する悪用のケースをテストおよびファジングするユニットテストや統合テストやファジングを構築することにより、各ビルドでこれらのコントロールが正しく動作していることをチェックしやすくなります。例えば、ログインコントローラのテストスイートとして、一般的なデフォルトユーザ名、アカウント列挙、総当たり攻撃、LDAP と SQL インジェクション、XSS のユーザ名パラメータをテストする追加のテストを作成することができます。同様に、パスワードパラメータのテストには一般的なパスワード、パスワード長、null バイトインジェクション、パラメータの削除、XSS などを含める必要があります。

### セキュア開発トレーニングのために

ASVS はセキュアソフトウェアの特性を定義するためにも使用できます。多くの「セキュアコーディング」コースはコーディングのヒントがわずかにあるだけの単なるエシカルハッキングコースです。これは開発者がよりセキュアなコードを書くのに必ずしも役に立つとは限りません。代わりに、セキュア開発コースでは、してはいけないことの Top 10 のような否定的なことではなく、ASVS にある肯定的なメカニズムに重点を置いて ASVS を使用できます。ASVS の構造は、アプリケーションを保護する際に、さまざまなトピックを順を追って説明するための論理的な構造も提供します。

### セキュアなソフトウェアの調達をガイドするためのフレームワークとして

ASVS は、セキュアなソフトウェアの調達やカスタム開発サービスの調達を支援する優れたフレームワークです。調達者は単に入手したいソフトウェアを ASVS レベル X で開発しなければならないという要件を設定し、そのソフトウェアが ASVS レベル X を満たすことを販売者に証明するよう要求できます。

## 実際に ASVS を適用する

脅威が異なれば動機も異なります。一部の業界では独自の情報資産と技術資産があり、ドメイン固有の規制順守要件があります。

組織はそのビジネスの性質に基づく独自のリスク特性を詳細に検討し、そのリスクとビジネス要件に基づいて適切な ASVS レベルを決定することを強く推奨します。

# 監査と認証

## ASVS 認証と認証マークに対する OWASP の見解

OWASP はベンダ中立の非営利組織であり、ベンダ、検証者、ソフトウェアの認証は行っていません。ASVS 準拠を主張する保証、認証マーク、認証はいずれも OWASP によって公式に承認されたものではないため、組織は第三者が主張する ASVS 認証に注意する必要があります。

OWASP の公式な認証であると主張しない限り、組織は保証サービスを提供できます。

## ASVS コンプライアンスの検証方法

ASVS はテストガイドのレベルでのコンプライアンスを検証する厳密な方法について意図的に明確にしていません。しかし、いくつかの重要なポイントを強調しておくことは重要です。

### 検証レポート

従来のペネトレーションテストは「例外による」問題を報告し、不合格のみをリストします。しかし、ASVS 認証レポートには、スコープ、チェックしたすべての要件の要約、例外が記録された要件、問題解決のガイダンスを含めるべきです。要件の中には適用できないもの (ステートレス API でのセッション管理など) もあり、その旨をレポートに記載しなければなりません。

### 検証のスコープ

一部の要件はアプリケーションの機能にとって無関係またはそれほど重要ではないため、アプリケーションを開発する組織は一般的にすべての要件を実装するわけではありません。検証者は、組織が達成しようとしているレベルや含まれている要件など、検証のスコープを明確に定義する必要があります。これは、含まれていないものではなく、含んでいるものの観点で行う必要があります。また、実施されていない要件を除外する根拠についての意見を示す必要もあります。

これにより、検証報告書の利用者は検証の背景を理解し、アプリケーションにおける信頼レベルについて十分に理解した上で判断できるようになります。

認証機関は適切なテスト手法を選択できますが、レポートでそれを開示する必要があり、理想的には再現できる必要があります。アプリケーションや要件に応じて、手動ペネトレーションテストやソースコード解析などのさまざまな手法を使用して、入力バリデーションなどの側面を検証できます。

### 検証メカニズム

特定の ASVS 要件を検証するには、さまざまな技法が必要になることがあります。ペネトレーションテスト (アプリケーションを完全にカバーするために有効なクレデンシャルを使用する) 以外にも、ASVS 要件を検証するには、ドキュメント、ソースコード、構成、および開発プロセスに関与する人々へのアクセスが必要となることがあります。特に L2 および L3 の要件を検証する場合、作業書類、スクリーンショット、スクリプト、テストログなどの詳細なドキュメントで、調査結果の堅実な証跡を提供することは標準的な慣行です。各要件は検証可能な形でテストしていなければならないため、徹底的なテストを行わずに自動化ツールを実行するだけでは認証には不十分です。

ASVS 要件の検証における自動化の使用は、常に関心を集めるトピックです。したがって、自動テストとブラックボックステストに関連するいくつかのポイントを明確にすることが重要です。

#### 自動セキュリティテストツールの役割

動的および静的アプリケーションセキュリティテストツール (DAST や SAST) などの自動セキュリティテストツールは、ビルドパイプラインに適切に実装されると、存在してはいけないセキュリティ問題を効果的に特定できるかもしれません。ただし、慎重な設定とチューニングを行わないと、必要なカバレッジが得られず、ノイズのレベルによって実際のセキュリティ問題を特定および緩和できなくなります。

これによって、出力エンコーディングやサニタイゼーションに関連するものなどの、より基本的で簡単な技術要件をカバーできるかもしれませんが、これらのツールではより複雑な ASVS 要件や、ビジネスロジックやアクセス制御に関連する要件の多くを完全には検証できないことに注意することが重要です。

それほど単純ではない要件では、自動化を利用できる可能性はありますが、これを実現するにはアプリケーション固有の検証を記述する必要があるでしょう。これらは組織がすでに使用している単体テストや統合テストに似ているかもしれません。そのため、この既存のテスト自動化インフラストラクチャを使用して、ASVS 固有のテストを記述することが可能かもしれません。これを行うには短期的な投資が必要ですが、これらの ASVS 要件を継続的に検証できるようになるという長期的な恩恵が大きくなるでしょう。

要約すると、自動化ツールを使用してテスト可能であることは、既製ツールを実行することではありません。

#### ペネトレーションテストの役割

バージョン 4.0 の L1 は「ブラックボックス」(ドキュメントもソースもない) テストを実行するように最適化されていましたが、それでも標準では、これは効果的な保証活動ではなく、積極的に推奨すべきではないことは明らかでした。

必要な追加情報にアクセスできない状態でテストを行うことは、セキュリティ検証のメカニズムとしては非効率的で非効果的です。それは、ソースをレビューし、脅威や対策漏れを特定し、より短い時間枠でのはるかに徹底的なテストを実施する可能性を逃すことになるためです。

従来のペネトレーションテストではなく、アプリケーション開発者とアプリケーションのドキュメントに完全にアクセスできる、ドキュメントやソースコード主導 (ハイブリッド) のペネトレーションテストを実施することを強くお勧めします。これは、多くの ASVS 要件を検証するために必要になるでしょう。

# v4.x と比較した変更点

## はじめに

バージョン 4.x の標準に慣れ親しんだユーザは、コンテンツ、スコープ、基本理念の更新など、バージョン 5.0 で導入された主な変更点を確認することが役立つかもしれません。

バージョン 4.0.3 の 286 の要件のうち、変更されていないのは 11 件のみであり、15 件はその意味を変えずに文法上の軽微な調整を行いました。計 109 件 (38%) の要件がバージョン 5.0 では独立した要件ではなくなりました。そのうち 50 件は単純に削除され、28 件は重複として削除され、31 件は他の要件に統合されました。残りは何らかの形で改訂されています。実質的に変更されていない要件であっても、順序変更や構造変更により識別子が異なります。

バージョン 5.0 の採用を促進するために、バージョン 4.x の要件がバージョン 5.0 の要件とどのように対応しているかをユーザが追跡できるように、マッピングドキュメントが提供されています。これらのマッピングはリリースのバージョン管理に縛られるものではなく、必要に応じて更新または明確化される可能性があります。

## 要件の理念

### スコープとフォーカス

バージョン 4.x では標準の意図するスコープに合致しない要件を含みましたが、これらは削除されました。また、5.0 のスコープ基準を満たしていない要件や検証不可能な要件も除外されました。

### メカニズムよりもセキュリティ目標を重視

バージョン 4.x では、多くの要件が基盤となるセキュリティ目標ではなく、特定のメカニズムに重点を置いていました。バージョン 5.0 では、セキュリティ目標を中心に据えて、特定のメカニズムについてはそれが唯一の現実的な解決策である場合にのみ言及するか、例や補足的なガイダンスとして提供します。

このアプローチは、特定のセキュリティ目標を達成するために複数の方法が存在しうることを認識し、組織の柔軟性を制限しかねない不必要な規範性を回避するものです。

さらに、同じセキュリティ上の懸念に対処する要件は、必要に応じて統合されています。

### セキュリティ上の決定事項の文書化

セキュリティ上の決定事項の文書化という概念はバージョン 5.0 での新たなもののように見えるかもしれませんが、これは以前のバージョン 4.0 におけるポリシー適用と脅威モデリングに関する要件の進化形です。以前は、許可されるネットワーク接続の決定など、セキュリティコントロールの実装に必要な分析を暗黙的に要求する要件もありました。

実装と検証に必要な情報が利用できるようにするため、これらの期待事項はドキュメント要件として明示的に定義され、明確で実行可能かつ検証可能になりました。

## 構造変更と新章

バージョン 5.0 では、いくつかの章に全く新しいコンテンツを導入しています。

* OAuth と OIDC – アクセス委譲やシングルサインオンのためにこれらのプロトコルが広く採用されていることから、開発者が遭遇する可能性のある多様なシナリオに対応するために専用の要件が追加されました。この分野は以前のバージョンにおけるモバイルおよび IoT の要件の扱いと同様に、最終的には独立した標準へと進化する可能性があります。
* WebRTC – このテクノロジが普及するにつれて、その固有のセキュリティ上の考慮事項と課題は専用のセクションで取り上げるようにしました。

また、各章と各セクションは、関連する要件の一貫したセットとして構成されるように、努力してきました。

この構造変更により、追加の章が設けられました。

* 自己完結型トークン – 以前はセッション管理に分類されていた自己完結型トークンは、現在では独立したメカニズムとして認識され、ステートレス通信 (OAuth や OIDC など) の基盤要素となっています。その固有のセキュリティ上の意味合いから、バージョン 5.x で導入されたいくつかの新しい要件とともに、専用の章で扱われます。
* Web フロントエンドセキュリティ – ブラウザベースのアプリケーションの複雑さが増したことと、API のみのアーキテクチャの台頭により、フロントエンドのセキュリティ要件は専用の章に分割されました。
* セキュアコーディングとアーキテクチャ – 既存の章に収まらない一般的なセキュリティプラクティスに対処する新しい要件がここにまとめられています。

バージョン 5.0 では意図を明確にするためにその他の構成変更も行われました。たとえば、入力バリデーション要件は、サニタイゼーションやエンコーディングと一緒に分類されるのではなく、ビジネスルールを強制する役割を反映して、ビジネスロジックと一緒とするように移動されました。

以前の V1 アーキテクチャの章は削除されました。その最初のセクションにはスコープ外の要件を含みましたが、後続のセクションは関連する章に再配置され、必要に応じて重複を排除し、明確化しました。

## 他の標準への直接マッピングの削除

他の標準への直接マッピングは、この標準の本体から削除されました。その目的は OWASP Common Requirement Enumeration (CRE) プロジェクトでマッピングを準備することであり、ASVS をさまざまな OWASP プロジェクトや外部標準とリンクします。

以下で説明するように、CWE と NIST への直接マッピングは維持されなくなりました。

### NIST Digital Identity Guidelines との結びつきの低減

NIST [Digital Identity Guidelines (SP 800-63)](https://pages.nist.gov/800-63-3/) は長年にわたり認証と認可コントロールのリファレンスとして使用されてきました。バージョン 4.x では、一部の章が NIST の構成と用語に密接に整合していました。

これらのガイドラインが重要なリファレンスであることに変わりはありませんが、厳密な整合はあまり広く認識されていない用語、類似した要件の重複、不完全なマッピングなどの課題をもたらしました。バージョン 5.0 ではこのアプローチから脱却し、明確性と関連性を向上します。

### 共通脆弱性タイプ一覧 (Common Weakness Enumeration, CWE) からの脱却

[Common Weakness Enumeration (CWE)](https://cwe.mitre.org/) はソフトウェアのセキュリティ脆弱性の有用な分類法を提供します。しかし、カテゴリのみの CWE、要件を単一の CWE にマッピングすることの難しさ、バージョン 4.x での不正確なマッピングの存在などの課題から、バージョン 5.0 では CWE への直接マッピングを廃止することを決定しました。

## レベル定義の再考

バージョン 4.x では、レベルを L1 (「最小 (Minimum)」)、L2 (「標準 (Standard)」)、L3 (「上級 (Advanced)」) と定義し、機密データを扱うすべてのアプリケーションは少なくとも L2 を満たす必要があると示唆していました。

バージョン 5.0 では、このアプローチにおけるいくつかの問題に対処しています。以下で説明します。

実際問題として、バージョン 4.x ではレベル指標にチェックマークを使用していましたが、5.x ではマークダウン、PDF、DOCX、CSV、JSON、XML を含む標準のすべての形式で単純な数字を使用しています。後方互換性のため、引き続きチェックマークを使用する CSV、JSON、XML 出力のレガシーバージョンも生成しています。

### より容易なエントリレベル

フィードバックによると、レベル 1 の要件数の多さ (約 120) に加え、ほとんどのアプリケーションには不十分な「最小 (Minimum)」レベルという指定が、採用を阻害していることが示されました。バージョン 5.0 では、レベル 1 を主に第一層の防御要件を中心に定義することで、この障壁を低くし、このレベルの要件をより明確で少なくすることを目指しています。これを数値で表すと、v4.0.3 では、計 278 件の要件のうち L1 要件は 128 件あり、46% を占めていました。5.0.0 では、計 345 件の要件のうち L1 要件は 70 件あり、20% を占めています。

### 試験性の誤り

バージョン 4.x でレベル 1 のコントロールを選定する際の重要な要素は、「ブラックボックス」外部ペネトレーションテストによる評価に適しているかどうかでした。しかし、このアプローチはセキュリティコントロールの最小セットとしてのレベル 1 の意図と完全には合致していませんでした。レベル 1 はアプリケーションを保護するには不十分であると主張するユーザもいれば、テストが難しすぎるというユーザもいました。

試験性を基準として頼ることは、相対的であり、時には誤解を招きます。要件がテスト可能であるという事実は、それが自動化された方法や簡単な方法でテストできることを保証するものではありません。さらに、最も簡単にテスト可能な要件が、必ずしもセキュリティへの影響が最も大きい要件や実装が最も簡単な要件とは限りません。

そのため、バージョン 5.0 では、レベルの決定は主にリスクの軽減に基づいて行われ、実装の労力も考慮しました。

### リスクだけではない

特定のアプリケーションに対して特定のレベルを義務付ける、規範的なリスクベースのレベルの使用は、過度に厳格であることが判明しています。実際には、セキュリティコントロールの優先順位付けと実装は、リスク軽減と実装に必要な労力など、複数の要因に依存します。

したがって、組織はその成熟度とユーザに送りたいメッセージに基づいて、達成すべきと思われるレベルを達成することが推奨されます。

# V1 エンコーディングとサニタイゼーション

## 管理目標

この章では、信頼できないデータの安全でない処理に関する、最も一般的な Web アプリケーションセキュリティの弱点を取り上げます。このような弱点は、信頼できないデータが、関連するインタプリタの構文規則を使用して解釈され、さまざまな技術的な脆弱性につながります。

現代の Web アプリケーションでは、パラメータ化クエリ、自動エスケープ、テンプレートフレームワークなどのより安全な API を使用することが常に最善です。それ以外には、出力のエンコード、エスケープ、サニタイズを注意深く実行することが、アプリケーションのセキュリティにとって重要になります。

入力バリデーションは、予期しないコンテンツや危険なコンテンツから保護するための多層防御メカニズムとして機能します。しかし、その主な目的は、入力されるコンテンツが機能上およびビジネス上の期待に一致することを確保することであるため、これに関する要件は「バリデーションとビジネスロジック」の章にあります。

## V1.1 エンコーディングおよびサニタイゼーションアーキテクチャ

下記のセクションでは、セキュリティの脆弱性を回避するために、安全でないコンテンツを安全に処理するための構文固有またはインタプリタ固有の要件を示します。このセクションの要件は、この処理が行われるべき順序と場所をカバーしています。また、二重エンコーディング問題を防ぐために、データが保存されるときは常に、エンコードまたはエスケープされた状態 (HTML エンコーディングなど) で保存するのではなく、元の状態を維持することを保証することも目的としています。

| # | 説明 | レベル |
| --- | --- | --- |
| **1.1.1** | 入力は一度だけ標準形式にデコードまたはアンエスケープされ、その形式でエンコードされたデータが期待される場合にのみデコードされ、これは入力をさらに処理する前に行われる。たとえば、入力バリデーションやサニタイゼーションの後には実行されない。 | 2 |
| **1.1.2** | アプリケーションはそれが意図されているインタプリタまたはインタプリタ自体によって使用される前の最終ステップとして、出力エンコーディングおよびエスケープを実行する。 | 2 |

## V1.2 インジェクション防御

潜在的に危険なコンテキストの近くあるいは隣接して実行される出力エンコードやエスケープは、あらゆるアプリケーションのセキュリティにとって重要です。通常、出力エンコードとエスケープは永続化されるのではなく、適切なインタプリタですぐに使用できるように出力を安全にするために使用されます。これをあまりに早い段階で行おうとすると、不正なコンテンツとなったり、エンコードやエスケープが効かなくなることがあります。

多くの場合、ソフトウェアライブラリは自動的にこれを行う安全な関数やより安全な関数を含みますが、それらが現在のコンテキストに対して正しいことを確認する必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **1.2.1** | HTTP レスポンス、HTML ドキュメント、XML ドキュメントの出力エンコーディングは、メッセージやドキュメント構造の変更を避けるために、HTML 要素、HTML 属性、HTML コメント、CSS、HTTP ヘッダフィールドに関連する文字をエンコードするなど、要求されるコンテキストに関連している。 | 1 |
| **1.2.2** | URL を動的に構築する場合、信頼できないデータはそのコンテキストに応じてエンコードされている (例: クエリやパスパラメータの URL エンコーディングや base64url エンコーディング)。安全な URL プロトコルのみが許可されるようにしている (例: javascript: や data: を許可しない)。 | 1 |
| **1.2.3** | 出力エンコーディングまたはエスケープは、JavaScript コンテンツ (JSON を含む) を動的に構築するときに使用され、メッセージやドキュメント構造の変更を回避している (JavaScript および JSON インジェクションを回避するため)。 | 1 |
| **1.2.4** | データ選択またはデータベースクエリ (SQL, HQL, NoSQL, Cypherなど) がパラメータ化クエリ、ORM、エンティティフレームワークもしくは他の方法により保護されており、SQL インジェクションや他のデータベースインジェクション攻撃の影響を受けない。これはストアドプロシージャを記述する際にも関係する。 | 1 |
| **1.2.5** | アプリケーションが OS コマンドインジェクションに対して保護していること、およびオペレーティングシステムコールがパラメータ化された OS クエリを使用するか、コンテキストに応じたコマンドライン出力エンコーディングを使用する。 | 1 |
| **1.2.6** | アプリケーションが LDAP インジェクション脆弱性に対して保護している、または LDAP インジェクションを防ぐために特定のセキュリティ管理策が実装されている。 | 2 |
| **1.2.7** | アプリケーションは、クエリパラメータ化やコンパイル済みクエリを使用することで、XPath インジェクション攻撃から保護されている。 | 2 |
| **1.2.8** | LaTeX プロセッサは ("--shell-escape" フラグを使用しないなど) 安全に構成されており、LaTeX インジェクション攻撃を防ぐためにコマンドの許可リストを使用している。 | 2 |
| **1.2.9** | アプリケーションは正規表現内の特殊文字をエスケープ (通常はバックスラッシュを使用) し、メタ文字として誤って解釈されることを防いでいる。 | 2 |
| **1.2.10** | アプリケーションが CSV インジェクションや数式インジェクションから保護されている。アプリケーションは CSV コンテンツをエクスポートする際に RFC 4180 セクション 2.6 および 2.7 で定義されているエスケープ規則に従う必要がある。さらに、CSV やその他のスプレッドシート形式 (XLS, XLSX, ODF など) をエクスポートする際に、特殊文字 ('=', '+', '-', '@', '\t' (タブ), '\0' (ヌル文字) など) がフィールド値の最初の文字として現れる場合、シングルクォートでエスケープする必要がある。 | 3 |

注: パラメータ化クエリの使用や SQL のエスケープだけでは必ずしも十分ではありません。テーブル名やカラム名 ("ORDER BY" 句のカラム名を含む) などのクエリ部分はエスケープできません。これらのフィールドにエスケープされたユーザ提供データを含めると、クエリの失敗や SQL インジェクションを引き起こします。

## V1.3 サニタイゼーション

信頼できないコンテンツを安全でないコンテキストで使用することに対する理想的な保護は、コンテキスト固有のエンコードやエスケープを使用することです。これにより、安全でないコンテンツと同じ意味を維持しながら、前のセクションで詳しく説明したように、この特定のコンテキストでの使用が安全になります。

これが不可能な場合、潜在的に危険な文字やコンテンツを削除するサニタイズが必要になります。場合によっては入力の意味を変えてしまうかもしれませんが、セキュリティ上の理由から、選択肢がないこともあります。

| # | 説明 | レベル |
| --- | --- | --- |
| **1.3.1** | WYSIWYG エディタなどから取得した信頼できない HTML 入力はすべて、よく知られた安全な HTML サニタイゼーションライブラリもしくはフレームワークの機能を使用してサニタイズされている。 | 1 |
| **1.3.2** | アプリケーションが eval() や他の Spring Expression Language (SpEL) などの動的コード実行機能を使用しない。代替手段がない場合は、実行前に含まれるユーザ入力をサニタイズする必要がある。 | 1 |
| **1.3.3** | 潜在的に危険なコンテキストに渡されるデータは、事前にサニタイズして、このコンテキストにとって安全な文字だけを許可したり、長すぎる入力を切り詰めるなどの安全対策を実施している。 | 2 |
| **1.3.4** | ユーザが提供する Scalable Vector Graphics (SVG) スクリプト可能コンテンツは、スクリプトや foreignObject を含まないなど、アプリケーションにとって安全なタグや属性 (図形描画など) のみを含むように検証またはサニタイズされている。 | 2 |
| **1.3.5** | アプリケーションは、マークダウン、CSS や XSL スタイルシート、BBCode などのユーザが提供するスクリプト可能コンテンツまたは式テンプレート言語コンテンツをサニタイズまたは無効化している。 | 2 |
| **1.3.6** | アプリケーションは、信頼できないデータをプロトコル、ドメイン、パス、ポートの許可リストに照らして検証し、そのデータを使用して別のサービスを呼び出す前に潜在的に危険な文字をサニタイズすることで、サーバサイドリクエストフォージェリ (SSRF) 攻撃から保護している。 | 2 |
| **1.3.7** | アプリケーションは信頼できない入力に基づくテンプレートの作成を許可しないことで、テンプレートインジェクション攻撃から保護している。代替手段がない場合、テンプレート作成中に動的に含まれるすべての信頼できない入力はサニタイズまたは厳密に確認する必要がある。 | 2 |
| **1.3.8** | アプリケーションは Java Naming and Directory Interface (JNDI) クエリで使用する前に信頼できない入力を適切にサニタイズし、JNDI を安全に構成して JNDI インジェクション攻撃を防いでいる。 | 2 |
| **1.3.9** | アプリケーションは memcache に送信する前にコンテンツをサニタイズし、インジェクション攻撃を防いでいる。 | 2 |
| **1.3.10** | 使用時に、予期しないあるいは悪意のある方法で解決される可能性のあるフォーマット文字列は、処理される前にサニタイズされている。 | 2 |
| **1.3.11** | SMTP インジェクションや IMAP インジェクションから保護するため、アプリケーションがメールシステムに渡す前にユーザ入力をサニタイズしている。 | 2 |
| **1.3.12** | 正規表現に指数関数的なバックトラッキングを引き起こす要素がないことを検証している。また、信頼できない入力をサニタイズし、ReDoS 攻撃や Runaway Regex 攻撃を軽減している。 | 3 |

## V1.4 メモリ、文字列、アンマネージドコード

以下の要件は、安全でないメモリ使用に関するリスクに対処するもので、通常はアプリケーションがシステム言語またはアンマネージドコードを使用する場合に適用されます。

場合によっては、スタックのランダム化やデータ実行防止などのバッファオーバーフローの保護と警告を有効にし、安全でないポインタ、メモリ、フォーマット文字列、整数、文字列操作が見つかった場合にビルドを中断するコンパイラフラグを設定することで、これを実現できる可能性があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **1.4.1** | アプリケーションはメモリセーフな文字列、安全なメモリコピーおよびポインタ演算を使用して、スタック、バッファ、ヒープのオーバーフローを検出または防止する。 | 2 |
| **1.4.2** | 整数オーバーフローを防ぐために符号、範囲、および入力のバリデーション技法が使用されている。 | 2 |
| **1.4.3** | 動的に割り当てられたメモリとリソースが解放され、解放されたメモリへの参照やポインタが削除されるか null に設定されて、ダングリングポインタや use-after-free 脆弱性を防いでいる。 | 2 |

## V1.5 安全なデシリアライゼーション

保存または転送された表現から実際のアプリケーションオブジェクトへのデータの変換 (デシリアライゼーション) はこれまでさまざまなコードインジェクション脆弱性の原因となってきました。このような問題を回避するには、このプロセスを慎重かつ安全に実行することが重要です。

特に、特定のデシリアライズ手法はプログラミング言語やフレームワークのドキュメントにおいて安全でないことが確認されており、信頼できないデータを安全に扱うことはできません。使用する各メカニズムについて、慎重なデューデリジェンスを実行する必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **1.5.1** | アプリケーションが XML パーサーを制限的な構成を使用するよう構成し、XML 外部エンティティ (XML eXternal Entity, XXE) 攻撃を防止するために外部エンティティの解決などの安全でない機能を無効にする。 | 1 |
| **1.5.2** | 信頼できないデータのデシリアライゼーションでは、オブジェクトタイプの許可リストを使用したり、クライアント定義のオブジェクトタイプを制限するなど、安全な入力処理を強制してデシリアライゼーション攻撃を防いでいる。明示的に安全でないと定義されているデシリアライゼーションメカニズムは信頼できない入力では使用してはいけない。 | 2 |
| **1.5.3** | JSON 相互運用性の脆弱性や、さまざまな URI やファイルの解析動作がリモートファイルインクルージョン (RFI) やサーバサイドリクエストフォージェリ (SSRF) 攻撃で悪用されるような問題を回避するために、同じデータ型に対してアプリケーションで使用されるさまざまなパーサー (JSON パーサー、XML パーサー、URL パーサーなど) は一貫した方法で解析を実行し、同じエンコードメカニズムを使用する。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP LDAP Injection Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.html)
* [OWASP Cross Site Scripting Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)
* [OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html)
* [OWASP XML External Entity Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html)
* [OWASP Web Security Testing Guide: Client-Side Testing](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/11-Client-side_Testing)
* [OWASP Java Encoding Project](https://owasp.org/owasp-java-encoder/)
* [DOMPurify - Client-side HTML Sanitization Library](https://github.com/cure53/DOMPurify)
* [RFC4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files](https://datatracker.ietf.org/doc/html/rfc4180#section-2)

特にデシリアライゼーションやパースの問題の詳細情報はこちらを参照してください。

* [OWASP Deserialization Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html)
* [An Exploration of JSON Interoperability Vulnerabilities](https://bishopfox.com/blog/json-interoperability-vulnerabilities)
* [Orange Tsai - A New Era of SSRF Exploiting URL Parser In Trending Programming Languages](https://www.blackhat.com/docs/us-17/thursday/us-17-Tsai-A-New-Era-Of-SSRF-Exploiting-URL-Parser-In-Trending-Programming-Languages.pdf)

# V2 バリデーションとビジネスロジック

## 管理目標

この章の目的は、検証対象のアプリケーションが以下の上位目標を満たすことを確認することです。

* アプリケーションが受け取る入力はビジネスまたは機能上の期待と一致している。
* ビジネスロジックフローは正しい手順通りに処理され、省略できない。
* ビジネスロジックに自動攻撃を検知し防止する制限とコントロールを含んでいる。自動攻撃の例としては、連続的な少額の送金や一度に百万人の友人の追加がある。
* 高い価値を持つビジネスロジックフローにおいて悪用ケースや悪用する人を想定している。また、なりすまし、改竄、情報漏洩、権限昇格攻撃の対策を行っている。

## V2.1 バリデーションとビジネスロジックドキュメント

バリデーションとビジネスロジックドキュメントは、ビジネスロジックの制限、バリデーションルール、および組み合わされたデータ項目のコンテキストの一貫性を明確に定義して、アプリケーションに実装する必要があるものを明確にします。

| # | 説明 | レベル |
| --- | --- | --- |
| **2.1.1** | アプリケーションのドキュメントには、期待される構造に対するデータ項目の妥当性をチェックする方法についての入力バリデーションルールを定義している。これは、クレジットカード番号、電子メールアドレス、電話番号のような一般的なデータ形式のこともあれば、内部データ形式のこともある。 | 1 |
| **2.1.2** | アプリケーションのドキュメントには、地区と郵便番号が一致することをチェックするなど、組み合わされたデータ項目の論理的およびコンテキスト的な一貫性を検証する方法を定義している。 | 2 |
| **2.1.3** | ビジネスロジックの制限とバリデーションに対する期待は、ユーザーごと、およびアプリケーション全体にわたって文書化されている。 | 2 |

## V2.2 入力バリデーション

効果的な入力バリデーションコントロールは、アプリケーションが受け取ることを期待するデータの種類に関して、ビジネスまたは機能上の期待を強制します。これにより優れたデータ品質を確保して攻撃対象領域を減らします。但し、他のコンポーネントでデータを使用する場合や出力にデータを提示する場合、正しいエンコーディング、パラメータ化、サニタイゼーションを使用する必要性をなくしたり置き換わるわけではありません。

このコンテキストでは、「入力」は、HTML フォームフィールド、REST リクエスト、URL パラメータ、HTTP ヘッダフィールド、クッキー、ディスク上のファイル、データベース、外部 API など、さまざまなソースからもたらされる可能性があります。

ビジネスロジックコントロールでは特定の入力を 100 未満の数値であるかどうかをチェックするかもしれません。機能上の期待としては、数値が特定の閾値未満であるかどうかをチェックすることがあります。その数値は特定のループを実行する回数を制御しており、高い数値は過剰な処理と潜在的なサービス拒否状態につながる可能性があるためです。

スキーマバリデーションを明示的に義務付けてはいませんが、JSON や XML を使用する HTTP API やその他のインタフェースの完全なバリデーションをカバーするには、これが最も効果的なメカニズムかもしれません。

スキーマバリデーションについては以下の点に注意してください。

* JSON スキーマバリデーション仕様の「公開バージョン」は運用準備が整っていると考えられていますが、厳密には「安定版」ではありません。JSON スキーマバリデーションを使用する場合は、以下の要件のガイダンスとのギャップがないことを確認してください。
* 使用している JSON スキーマバリデーションライブラリも監視して、標準が正式に制定されたら、必要に応じて更新する必要があります。
* DTD に対する XXE 攻撃の問題を回避するため、DTD バリデーションは使用すべきではありません。また、フレームワークの DTD 評価を無効にする必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **2.2.1** | 入力はその入力に対するビジネスまたは機能上の期待を満たすことを検証されている。これは、許容される値、パターン、範囲のリストに対する肯定的なバリデーションを使用するか、あらかじめ定義されたルールに従って、期待される構造および論理的限界と入力との比較に基づく必要がある。L1 では、これは特定のビジネスまたはセキュリティ上の意思決定に使用される入力に焦点を当てることができる。L2 以上では、これはすべての入力に適用する必要がある。 | 1 |
| **2.2.2** | アプリケーションは信頼できるサービスレイヤで入力バリデーションを実行するように設計されている。クライアントサイドのバリエーションはユーザービリティを向上するため推奨されるが、セキュリティコントロールとしては依存してはいけない。 | 1 |
| **2.2.3** | アプリケーションは、関連するデータ項目の組み合わせが事前に定義されたルールに従って妥当であることを確保している。 | 2 |

## V2.3 ビジネスロジックのセキュリティ

このセクションでは、アプリケーションがビジネスロジックプロセスを正しい方法で実行し、アプリケーションのロジックとフローを悪用する攻撃に対して脆弱にならないようにするための主要な要件について検討します。

| # | 説明 | レベル |
| --- | --- | --- |
| **2.3.1** | アプリケーションは同じユーザのビジネスロジックフローを期待した正しい手順通り、省略せずに処理する。 | 1 |
| **2.3.2** | ビジネスロジックの制限はアプリケーションのドキュメントに従って実装され、ビジネスロジックの欠陥が悪用されることを避けている。 | 2 |
| **2.3.3** | トランザクションはビジネスロジックレベルで使用されてビジネスロジックオペレーション全体が成功するか、以前の正しい状態にロールバックされる。 | 2 |
| **2.3.4** | ビジネスロジックレベルのロックメカニズムを使用して、アプリケーションのロジックを操作することで、数量に限りのあるリソース (劇場の座席や配送枠など) が二重に予約されないようにしている。 | 2 |
| **2.3.5** | 価値の高いロジックフローは複数ユーザの承認を必要とし、不正なアクションや偶発的なアクションを防いでいる。これには、多額の送金、契約の承認、機密情報へのアクセス、製造における安全のオーバーライドなどが含まれるが、これらに限定されない。 | 3 |

## V2.4 アンチオートメーション

このセクションは、人間のようなインタラクションが要求され、過剰な自動化リクエストが防止されるようにするためのアンチオートメーションコントロールを含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **2.4.1** | アンチオートメーションコントロールが導入され、データ流出、ガベージデータ作成、クォータ枯渇、レート制限違反、サービス拒否、高価なリソースの過剰使用につながる可能性のあるアプリケーション機能への過剰な呼び出しから保護している。 | 2 |
| **2.4.2** | ビジネスロジックフローは現実的な人間のタイミングを必要とし、過度に迅速なトランザクション送信を防いでいる。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP Web Security Testing Guide: Input Validation Testing](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/README.html)
* [OWASP Web Security Testing Guide: Business Logic Testing](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/10-Business_Logic_Testing/README)
* [OWASP Automated Threats to Web Applications](https://owasp.org/www-project-automated-threats-to-web-applications/) の使用など、アンチオートメーションはさまざまな方法で実現できます。
* [OWASP Input Validation Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html)
* [JSON Schema](https://json-schema.org/specification.html)

# V3 Web フロントエンドセキュリティ

## 管理目標

このカテゴリは Web フロントエンドを介して実行される攻撃から保護する要件に焦点を当てています。これらの要件はマシン間ソリューションには適用されません。

## V3.1 Web フロントエンドセキュリティドキュメント

このセクションではアプリケーションのドキュメントに指定する必要があるブラウザのセキュリティ機能について概説します。

| # | 説明 | レベル |
| --- | --- | --- |
| **3.1.1** | アプリケーションドキュメントには、アプリケーションを使用するブラウザがサポートする必要がある想定されるセキュリティ機能 (HTTPS、HTTP Strict Transport Security (HSTS)、コンテンツセキュリティポリシー (CSP)、その他の関連する HTTP セキュリティメカニズムなど) を記載している。これらの機能の一部が利用できない場合にアプリケーションがどのように動作しなければならないか (ユーザへの警告やアクセスのブロックなど) も定義する必要がある。 | 3 |

## V3.2 意図しないコンテンツ解釈

コンテンツや機能を不適切なコンテンツでレンダリングすると、悪意のあるコンテンツが実行または表示される可能性があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **3.2.1** | ブラウザが不正なコンテキスト (API、ユーザがアップロードしたファイル、または他のリソースが直接リクエストされる場合など) で HTTP レスポンスのコンテンツや機能をレンダリングすることを防ぐために、セキュリティ制御が行われている。可能な制御には、HTTP リクエストヘッダフィールド (Sec-Fetch-\* など) が正しいコンテキストであることを示さない限りコンテンツを提供しないこと、Content-Security-Policy ヘッダフィールドの sandbox ディレクティブを使用するか、Content-Disposition ヘッダフィールドの attachment ディポジションタイプを使用することなどがある。 | 1 |
| **3.2.2** | HTML としてレンダリングするのではなく、テキストとして表示することを意図したコンテンツは、HTML や JavaScript などのコンテンツの意図しない実行を防ぐために、安全なレンダリング関数 (createTextNode や textContent など) を使用して処理している。 | 1 |
| **3.2.3** | アプリケーションは、明示的な変数宣言の採用、厳密な型チェックの実行、document オブジェクトへのグローバル変数の保存の回避、名前空間分離の実装によって、クライアントサイド JavaScript を使用する際の DOM clobbering を回避している。 | 3 |

## V3.3 クッキーセットアップ

このセクションでは、機密性の高いクッキーがアプリケーション自体によって作成されたことをより確実にし、その内容が漏洩したり不適切に変更されることを防ぐために、機密性の高いクッキーを安全に構成するための要件について概説します。

| # | 説明 | レベル |
| --- | --- | --- |
| **3.3.1** | クッキーには 'Secure' 属性が設定されており、クッキー名に '\_\_Host-' プレフィックスが使用されていない場合は、クッキー名に '\_\_Secure-' プレフィックスを使用する必要がある。 | 1 |
| **3.3.2** | 各クッキーの 'SameSite' 属性値はクッキーの目的に応じて設定され、一般にクロスサイトリクエストフォージェリ (CSRF) として知られる、ユーザーインタフェースのリドレス攻撃やブラウザベースのリクエストフォージェリ攻撃への露出を制限している。 | 2 |
| **3.3.3** | クッキーは、明示的に他のホストと共有するように設計されていない限り、クッキー名に '\_\_Host-' プレフィックスを付けている。 | 2 |
| **3.3.4** | クッキーの値がクライアントサイドスクリプトからアクセスされることを意図していない場合 (セッショントークンなど)、クッキーは 'HttpOnly' 属性が設定される必要があり、同じ値 (セッショントークンなど) は 'Set-Cookie' ヘッダフィールドを介してのみクライアントに転送される必要がある。 | 2 |
| **3.3.5** | アプリケーションがクッキーを書き込む際、クッキー名と値を合わせた長さは 4096 バイトを超えない。大きすぎるクッキーはブラウザに保存されないため、リクエストとともに送信されず、ユーザはそのクッキーに依存するアプリケーション機能を使用できなくなる。 | 3 |

## V3.4 ブラウザのセキュリティメカニズムヘッダ

このセクションでは、アプリケーションからのレスポンスを処理する際にブラウザのセキュリティ機能と制限を有効にするために、HTTP レスポンスに設定する必要があるセキュリティヘッダについて説明します。

| # | 説明 | レベル |
| --- | --- | --- |
| **3.4.1** | HTTP Strict Transport Security (HSTS) ポリシーを適用するために、Strict-Transport-Security ヘッダフィールドがすべてのレスポンスに含まれている。少なくとも 1 年間の最大有効期間を定義する必要があり、L2 以上では、ポリシーをすべてのサブドメインにも適用する必要がある。 | 1 |
| **3.4.2** | クロスオリジンリソース共有 (CORS) Access-Control-Allow-Origin ヘッダフィールドがアプリケーションによって固定値にしている。または、Origin HTTP リクエストヘッダフィールド値が使用される場合は、信頼できるオリジンの厳密な許可リストに対して検証されている。'Access-Control-Allow-Origin: \*' を使用する必要がある場合、レスポンスに機密情報が含まれていない。 | 1 |
| **3.4.3** | HTTP レスポンスには Content-Security-Policy ヘッダフィールドを含み、悪意のある JavaScript の実行を制限するために、ブラウザが信頼できるコンテンツやリソースのみをロードして実行するようにディレクトリを定義している。少なくとも、ディレクティブ object-src 'none' および base-uri 'none' を含み、許可リストを定義するか、ナンスまたはハッシュを使用するグローバルポリシーを使用する必要がある。L3 アプリケーションでは、ナンスまたはハッシュを用いたレスポンスごとのポリシーを定義する必要がある。 | 2 |
| **3.4.4** | すべての HTTP レスポンスには 'X-Content-Type-Options: nosniff' ヘッダフィールドを含んでいる。これは、与えられたレスポンスに対してコンテンツスニッフィングと MIME タイプ推測を使用しないようにブラウザに指示し、レスポンスの Content-Type ヘッダフィールド値が宛先リソースと一致することを要求します。たとえば、スタイルに対するリクエストへのレスポンスは、レスポンスの Content-Type が 'text/css' である場合にのみ受け入れられる。また、これによりブラウザの Cross-Origin Read Blocking (CORB) 機能の使用も可能にする。 | 2 |
| **3.4.5** | アプリケーションは 'Referer' HTTP リクエストヘッダフィールドを介して、技術的に機密データがサードパーティサービスに漏洩することを防ぐために、リファラポリシーを設定している。これは Referrer-Policy HTTP レスポンスヘッダフィールドを使用するか、HTML 要素属性を介して使用して行うことができる。機密データは URL 内のパスやクエリデータ、そして内部の非公開アプリケーションではホスト名も含むことがある。 | 2 |
| **3.4.6** | Web アプリケーションはすべての HTTP レスポンスに対して Content-Security-Policy ヘッダフィールドの frame-ancestors ディレクティブを使用して、デフォルトでは埋め込むことができないようにし、特定のリソースの埋め込みは必要な場合にのみ許可されるようにしている。X-Frame-Options ヘッダフィールドは、ブラウザによってサポートされているが、廃止されており、信頼できないことに注意する。 | 2 |
| **3.4.7** | Content-Security-Policy ヘッダフィールドが違反を報告する場所を指定している。 | 3 |
| **3.4.8** | ドキュメントのレンダリングを開始するすべての HTTP レスポンス (Content-Type text/html のレスポンスなど) には、必要に応じて、same-origin ディレクティブまたは the same-origin-allow-popups ディレクティブを持つ Cross‑Origin‑Opener‑Policy ヘッダフィールドを含んでいる。これにより、タブナビングやフレームカウンティングなど、Window オブジェクトへの共有アクセスを悪用する攻撃を防いでいる。 | 3 |

## V3.5 ブラウザのオリジン分離

サーバサイドで機密機能へのリクエストを受け入れる場合、アプリケーションはそのリクエストがアプリケーション自体あるいは信頼できるパーティによって開始され、攻撃者によって偽造されていないことを確認する必要があります。

このコンテキストにおける機密機能には、認証されたユーザと認証されていないユーザのフォーム投稿の受け入れ (認証リクエストなど)、状態変更操作、リソース要求機能 (データエクスポートなど) があります。

ここでの重要な保護は JavaScript の Same Origin Policy やクッキーの SameSite ロジックなどのブラウザセキュリティポリシーです。もう一つの一般的な保護は CORS プリフライトメカニズムです。このメカニズムは、異なるオリジンから呼び出されるように設計されたエンドポイントにとって重要ですが、異なるオリジンから呼び出されるように設計されていないエンドポイントにとっても、リクエストフォージェリ防止メカニズムとして役立ちます。

| # | 説明 | レベル |
| --- | --- | --- |
| **3.5.1** | アプリケーションが、機密機能を使用するための、許可されていないクロスオリジンリクエストを防ぐために、CORS プリフライトメカニズムに依存していない場合、これらのリクエストを検証して、アプリケーション自体から発信したものであることを確認している。これはフォージェリ防止トークンを使用して検証するか、CORS セーフリストリクエストヘッダフィールドではない追加の HTTP ヘッダフィールドを要求することで行われる。これは、一般にクロスサイトリクエストフォージェリ (CSRF) と呼ばれる、ブラウザベースのリクエストフォージェリ攻撃を防御するためのものである。 | 1 |
| **3.5.2** | アプリケーションが、機密機能の許可されていないクロスオリジン使用を防ぐために、CORS プリフライトメカニズムに依存している場合、CORS プリフライトリクエストをトリガーしないリクエストで機能を呼び出すことはできない。これは 'Origin' および 'Content-Type' リクエストヘッダフィールドの値をチェックするか、CORS セーフリストのヘッダフィールドではない追加のヘッダフィールドを使用する必要があるかもしれない。 | 1 |
| **3.5.3** | 機密機能の HTTP リクエストは POST, PUT, PATCH, DELETE などの適切な HTTP メソッドを使用し、HEAD, OPTIONS, GET などの HTTP 仕様で「安全」と定義されているメソッドを使用しない。あるいは、Sec-Fetch-\* リクエストヘッダフィールドの厳密なバリデーションを使用して、不適切なクロスオリジンコール、ナビゲーションリクエスト、期待されていないリソースロード (画像ソースなど) からリクエストが発生していないことを確保している。 | 1 |
| **3.5.4** | あるオリジンでロードされたドキュメントやスクリプトが別のオリジンのリソースとやり取りする方法や、クッキーのホスト名制限など、同一オリジンポリシーによって提供される制限を活用するために、別のアプリケーションは異なるホスト名でホストされている。 | 2 |
| **3.5.5** | メッセージのオリジンが信頼できない場合、またはメッセージの構文が無効な場合、postMessage インタフェースによって受信したメッセージを破棄している。 | 2 |
| **3.5.6** | クロスサイトスクリプトインクルージョン (XSSI) 攻撃を避けるために、JSONP 機能はアプリケーションのどの部分でも有効になっていない。 | 3 |
| **3.5.7** | クロスサイトスクリプトインクルージョン (XSSI) 攻撃を防ぐために、認可が必要なデータは JavaScript ファイルなどのスクリプトリソースレスポンスには含まれていない。 | 3 |
| **3.5.8** | 認証されたリソース (画像、動画、スクリプト、その他のドキュメントなど) は、意図した場合にのみユーザの代わりにロードまたは埋め込むことができる。これは、Sec-Fetch-\* HTTP リクエストヘッダフィールドの厳密なバリデーションにより、リクエストが不適切なクロスオリジンコールから生じたものではないことを確保するか、制限的な Cross-Origin-Resource-Policy HTTP レスポンスヘッダフィールドを設定することにより、ブラウザが返されたコンテンツをブロックするように指示することで実現できる。 | 3 |

## V3.6 外部リソース完全性

このセクションではサードパーティのサイトでコンテンツを安全にホストするためのガイダンスを提供します。

| # | 説明 | レベル |
| --- | --- | --- |
| **3.6.1** | JavaScript ライブラリ、CSS、Web フォントなどのクライアントサイド資産は、リソースが静的かつバージョン管理されており、サブリソース完全性 (Subresource Integrity, SRI) を使用して資産の完全性を検証している場合にのみ、外部 (コンテンツ配信ネットワーク (Content Delivery Network) など) にホストされている。これが不可能な場合は、各リソースについて、この正当性を証明するセキュリティ上の決定事項を文書化する必要がある。 | 3 |

## V3.7 ブラウザのセキュリティに関するその他の考慮事項

このセクションではクライアントサイドのブラウザセキュリティに必要となるその他のさまざまなセキュリティコントロールや最新のブラウザセキュリティ機能を含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **3.7.1** | アプリケーションは依然としてサポートされており安全であると考えられているクライアントサイドのテクノロジのみを使用している。この要件を満たさないテクノロジの例としては NSAPI プラグイン、Flash、Shockwave、ActiveX、Silverlight、NACL、クライアントサイド Java アプレットなどがある。 | 2 |
| **3.7.2** | アプリケーションは、宛先が許可リストに記載されている別のホスト名やドメイン (アプリケーションで制御されていない) にのみユーザーを自動的にリダイレクトしている。 | 2 |
| **3.7.3** | ユーザがアプリケーションの制御外の URL にリダイレクトされる場合、アプリケーションはナビゲーションをキャンセルするオプションとともに通知を表示している。 | 3 |
| **3.7.4** | アプリケーションのトップレベルドメイン (例: site.tld) が HTTP Strict Transport Security (HSTS) のパブリックプリロードリストに追加されている。これにより、アプリケーションの TLS の使用が、Strict-Transport-Security レスポンスヘッダフィールドのみに依存するのではなく、メインブラウザに直接組み込まれるようになる。 | 3 |
| **3.7.5** | アプリケーションへのアクセスに使用されるブラウザが、想定されるセキュリティ機能をサポートしていない場合、アプリケーションはドキュメントに記載されているとおりに動作する (ユーザへの警告やアクセスのブロックなど)。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [Set-Cookie \_\_Host- prefix details](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie#cookie_prefixes)
* [OWASP Content Security Policy Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html)
* [OWASP Secure Headers Project](https://owasp.org/www-project-secure-headers/)
* [OWASP Cross-Site Request Forgery Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)
* [HSTS Browser Preload List submission form](https://hstspreload.org/)
* [OWASP DOM Clobbering Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/DOM_Clobbering_Prevention_Cheat_Sheet.html)

# V4 API と Web サービス

## 管理目標

Web ブラウザや他のコンシューマが使用する API を公開 (通常 JSON, XML, GraphQL を使用) するアプリケーションには、特にいくつかの考慮すべき事項があります。この章では、関連するセキュリティ構成と適用すべきメカニズムについて説明します。

他の章にある認証、セッション管理、入力バリデーションの懸念事項は API にも適用されるので、この章をコンテキストから切り離したり、個別にテストすることはできないことに注意してください。

## V4.1 一般的な Web サービスセキュリティ

このセクションでは、一般的な Web サービスセキュリティの考慮事項と、その結果としての基本的な Web サービス衛生習慣について説明します。

| # | 説明 | レベル |
| --- | --- | --- |
| **4.1.1** | メッセージボディを持つすべての HTTP レスポンスにはレスポンスの実際のコンテンツと一致する Content-Type ヘッダフィールドを含んでいる。これには "text/", "/+xml", "/xml" などの IANA メディアタイプに従って安全な文字エンコーディング (UTF-8, ISO-8859-1 など) を指定する charset パラメータを含んでいる。 | 1 |
| **4.1.2** | ユーザ向けのエンドポイント (手動でのウェブブラウザアクセス用) のみが HTTP から HTTPS に自動的にリダイレクトし、他のサービスやエンドポイントでは透過的なリダイレクトを実装していない。これは、クライアントが誤って暗号化されていない HTTP リクエストを送信しても、リクエストが自動的に HTTPS に自動的にリダイレクトされるため機密データの漏洩が発見されない、という状況を避けるためである。 | 2 |
| **4.1.3** | アプリケーションにより使用され、ロードバランサ、Web プロキシ、backend-for-frontend サービスなどの中間層により設定される HTTP ヘッダフィールドはエンドユーザによって上書きできない。ヘッダの例としては X-Real-IP, X-Forwarded-\*, X-User-ID などがある。 | 2 |
| **4.1.4** | アプリケーションまたはその API によって明示的にサポートされている HTTP メソッド (preflight リクエスト時の OPTIONS を含む) のみが使用でき、未使用のメソッドはブロックされる。 | 3 |
| **4.1.5** | メッセージごとのデジタル署名を使用して、機密性が高いリクエストやトランザクションや、多数のシステムを横断するリクエストまたはトランザクションに対して、トランスポート保護の上にさらなる保証を提供している。 | 3 |

## V4.2 HTTP メッセージ構造バリデーション

このセクションでは、リクエストスマグリング、レスポンス分割、ヘッダインジェクション、長すぎる HTTP メッセージによるサービス拒否などの攻撃を防ぐために、HTTP メッセージの構造とヘッダフィールドを検証する方法について説明します。

これらの要件は、一般的な HTTP メッセージの処理と生成に関連しますが、異なる HTTP バージョン間で HTTP メッセージを変換する際に特に重要です。

| # | 説明 | レベル |
| --- | --- | --- |
| **4.2.1** | すべてのアプリケーションコンポーネント (ロードバランサ、ファイアウォール、アプリケーションサーバを含む) は、HTTP バージョンに適したメカニズムを使用して、受信 HTTP メッセージの境界を決定し、HTTP リクエストスマグリングを防いでいる。HTTP/1.x では、Transfer-Encoding ヘッダフィールドが存在する場合、Content-Length ヘッダフィールドを RFC 2616 に従って無視する必要がある。HTTP/2 または HTTP/3 を使用する際に、Content-Length ヘッダフィールドが存在する場合、受信側はそれが DATA フレームの長さと一致することを確保する必要がある。 | 2 |
| **4.2.2** | HTTP メッセージを生成する際、HTTP リクエストスマグリングを防ぐために、Content-Length ヘッダフィールドが HTTP プロトコルのフレームで決定されるコンテンツの長さと競合しない。 | 3 |
| **4.2.3** | アプリケーションは、レスポンス分割やヘッダインジェクション攻撃を防ぐために、Transfer-Encoding などの接続固有のヘッダフィールドを持つ HTTP/2 や HTTP/3 メッセージを送受信しない。 | 3 |
| **4.2.4** | アプリケーションは、ヘッダインジェクション攻撃を防ぐために、ヘッダフィールドと値に CR (\r)、LF (\n)、CRLF (\r\n) シーケンスを含まない HTTP/2 および HTTP/3 リクエストのみを受け付ける。 | 3 |
| **4.2.5** | アプリケーション (バックエンドまたはフロントエンド) がリクエストを構築して送信する場合、バリデーション、サニタイゼーション、またはその他のメカニズムを使用して、受信コンポーネントで受け入れられるには長すぎる URI (API 呼び出しのためなど) や HTTP リクエストヘッダフィールド (認可やクッキーのためなど) の作成を回避している。長すぎるリクエスト (長いクッキーヘッダフィールドなど) を送信すると、サーバが常にエラーステータスで応答するなど、サービス拒否を引き起こす可能性がある。 | 3 |

## V4.3 GraphQL

GraphQL はさまざまなバックエンドサービスと密結合しないデータリッチなクライアントを作成する方法として、より一般的になりつつあります。このセクションでは GraphQL のセキュリティに関する考慮事項について説明します。

| # | 説明 | レベル |
| --- | --- | --- |
| **4.3.1** | コストが高く、ネストされたクエリの結果としての GraphQL やデータレイヤエクスプレッションのサービス拒否 (DoS) を防止するために、クエリ許可リスト、深さ制限、量制限、またはクエリコスト分析を使用している。 | 2 |
| **4.3.2** | Graph API が他の関係者により使用されることを意図している場合を除き、本番環境では GraphQL イントロスペクションクエリが無効になっている。 | 2 |

## V4.4 WebSocket

WebSocket は単一の TCP 接続で同時双方向通信チャネルを提供する通信プロトコルです。2011 年に IETF によって RFC 6455 として標準化され、HTTP ポート 443 と 80 で動作するように設計されていますが、HTTP とは異なります。

このセクションでは、特にこのリアルタイム通信チャネルを悪用する、通信セキュリティとセッション管理に関連する攻撃を防ぐための主要なセキュリティ要件について説明します。

| # | 説明 | レベル |
| --- | --- | --- |
| **4.4.1** | WebSocket over TLS (WSS) がすべての WebSocket 接続に使用されている。 | 1 |
| **4.4.2** | 最初の HTTP WebSocket ハンドシェイクで Origin ヘッダフィールドがアプリケーションで許可されているオリジンのリストと照合されている。 | 2 |
| **4.4.3** | アプリケーションの標準セッション管理を使用できない場合、関連するセッション管理セキュリティ要件に準拠した専用トークンが使用されている。 | 2 |
| **4.4.4** | 専用の WebSocket セッション管理トークンは、既存の HTTPS セッションを WebSocket チャネルに移行するときに、以前に認証された HTTPS セッションを通じて最初に取得または検証されます。 | 2 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP REST Security Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html)
* [graphql.org](https://graphql.org/learn/authorization/) と [Apollo](https://www.apollographql.com/docs/apollo-server/security/authentication/#authorization-methods) の GraphQL Authorization のリソース
* [OWASP Web Security Testing Guide: GraphQL Testing](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/12-API_Testing/01-Testing_GraphQL)
* [OWASP Web Security Testing Guide: Testing WebSockets](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/11-Client-side_Testing/10-Testing_WebSockets)

# V5 ファイル処理

## 管理目標

ファイルを使用すると、サービス拒否、認可されていないアクセス、ストレージ枯渇など、アプリケーションにさまざまなリスクをもたらす可能性があります。この章にはこれらのリスクに対処するための要件を含みます。

## V5.1 ファイル処理ドキュメント

このセクションは、関連するセキュリティチェックを開発および検証するために必要な事前条件として、アプリケーションに受け入れられるファイルに期待される特性を文書化する要件を含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **5.1.1** | ドキュメントでは各アップロード機能で許可されるファイルタイプ、想定されるファイル拡張子、最大サイズ (展開後のサイズを含む) を定義している。さらに、悪意のあるファイルが検出された場合にアプリケーションがどのように動作するかなど、ドキュメントにはエンドユーザがファイルをダウンロードして処理する際に安全を確保する方法を指定している。 | 2 |

## V5.2 ファイルアップロードとコンテンツ

ファイルアップロード機能は信頼できないファイルの主な発生源です。このセクションでは、これらのファイルの存在、量、内容がアプリケーションに悪影響を及ぼさないようにするための要件について説明します。

| # | 説明 | レベル |
| --- | --- | --- |
| **5.2.1** | アプリケーションはパフォーマンスの低下やサービス拒否攻撃を引き起こすことなく処理可能なサイズのファイルのみを受け付けている。 | 1 |
| **5.2.2** | アプリケーションはファイルを受け入れる際に、それ自体または zip ファイルなどのアーカイブ内のいずれかで、ファイル拡張子が予期されるファイル拡張子と一致するかどうかをチェックし、内容がその拡張子で表されるタイプに対応しているかどうかを検証している。これには最初の 'マジックバイト' のチェック、イメージの再書き込みの実行、ファイル内容のバリデーションのための専用ライブラリの使用が含まれるがこれに限定されない。L1 では、特定のビジネスまたはセキュリティ上の決定を行うために使用されるファイルのみに焦点を当てることができる。L2 以上では、受け入れられるすべてのファイルに適用する必要がある。 | 1 |
| **5.2.3** | アプリケーションが圧縮ファイル (zip, gz, docx, odt など) を展開する前に最大許容非圧縮サイズおよび最大ファイル数と照合している。 | 2 |
| **5.2.4** | 一人のユーザがあまりにも多くのファイルや極端に大きいファイルでストレージを圧迫させることができないように、ユーザあたりのファイルサイズクォータと最大ファイル数が適用されている。 | 3 |
| **5.2.5** | 特に必要な場合 (その場合、シンボリックリンクできるファイルの許可リストを適用する必要がある) を除き、アプリケーションはシンボリックリンクを含む圧縮ファイルのアップロードを許可していない。 | 3 |
| **5.2.6** | アプリケーションはピクセルフラッド攻撃を防ぐために、許容される最大値を超えるピクセルサイズのアップロードされた画像を拒否している。 | 3 |

## V5.3 ファイル保存

このセクションは、アップロード後にファイルが不適切に実行されるのを防ぎ、危険なコンテンツを検出し、信頼できないデータを使用してファイルが保存される場所を制御するのを防ぐための要件を含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **5.3.1** | 信頼できない入力によってアップロードまたは生成され、パブリックフォルダに保存されたファイルは、HTTP リクエストで直接アクセスした場合、サーバサイドのプログラムコードとして実行できない。 | 1 |
| **5.3.2** | アプリケーションがファイル操作のためにファイルパスを作成する際、ユーザが送信したファイル名の代わりに、内部で生成されたデータまたは信頼できるデータを使用する。また、ユーザが送信したファイル名やファイルメタデータを使用する必要がある場合は、厳密なバリデーションとサニタイゼーションを適用する必要がある。これは、パストラバーサル、ローカルまたはリモートファイルインクルージョン (LFI, RFI)、サーバサイドリクエストフォージェリ (SSRF) 攻撃から保護するためである。 | 1 |
| **5.3.3** | zip スリップなどの脆弱性を防ぐために、ファイル展開などのサーバサイドのファイル処理が、ユーザから提供されたパス情報を無視する。 | 3 |

## V5.4 ファイルダウンロード

このセクションは、パストラバーサルやインジェクション攻撃など、ダウンロードするファイルを提供する際のリスクを提供する際のリスクを緩和するための要件を含みます。また、これには危険なコンテンツを含まないことを確認することも含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **5.4.1** | アプリケーションが JSON、JSONP、URL パラメータに含まれるユーザが送信したファイル名を確認または無視し、レスポンスの Content-Disposition ヘッダフィールドでファイル名を指定している。 | 2 |
| **5.4.2** | 提供されるファイル名 (HTTP レスポンスヘッダフィールドや電子メールの添付ファイルなど) は、ドキュメント構造を保持し、インジェクション攻撃を防ぐために、エンコードまたはサニタイズされている (RFC 6266 に従うなど)。 | 2 |
| **5.4.3** | 信頼できない場所から取得したファイルが、既知の悪意のあるコンテンツの配信を防ぐためにアンチウィルススキャナで検査されている。 | 2 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP File Upload Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html)
* [Example of using symlinks for arbitrary file read](https://hackerone.com/reports/1439593)
* [Explanation of "Magic Bytes" from Wikipedia](https://en.wikipedia.org/wiki/List_of_file_signatures)

# V6 認証

## 管理目標

認証とは個人またはデバイスの真正性を確立または確認するプロセスです。これには個人またはデバイスに関する主張を検証し、なりすましへの耐性を確保し、パスワードのリカバリや傍受を防止することが含まれます。

[NIST SP 800-63](https://pages.nist.gov/800-63-3/) は世界中の組織にとって価値のある最新のエビデンスベースの標準ですが、特に米国の機関および米国の機関とやり取りする組織に関連します。

この章の要件の多くは標準 (NIST SP 800-63B "Digital Identity Guidelines - Authentication and Lifecycle Management") の二番目のセクションに基づいていますが、この章では一般的な脅威と頻繁に悪用される認証の弱点に焦点を当てており、標準のすべてのポイントを包括的に網羅するものではありません。NIST SP 800-63 への完全な準拠が必要な場合は、NIST SP 800-63 を参照してください。

さらに、NIST SP 800-63 の用語は時として異なることがあり、この章ではより一般的に理解されている用語を使用することで、わかりやすさを向上します。

より高度なアプリケーションに共通する機能の一つは、さまざまなリスク要因に基づいて、必要な認証ステージを適応する機能です。これらのメカニズムも認可の決定に考慮される必要があるため、この機能は「認可」の章でカバーします。

## V6.1 認証ドキュメント

このセクションはアプリケーションについて保持すべき認証ドキュメントの詳細を規定する要件を含みます。これは関連する認証コントロールをどのように構成すべきかを実装して評価する上で非常に重要です。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.1.1** | アプリケーションドキュメントでは、レート制限、自動化防止、適用応答などのコントロールが、クレデンシャルスタッフィングやパスワードブルートフォースなどの攻撃から防御するためにどのように使用されるかを定義している。このドキュメントでは、これらのコントロールがどのように構成されるかを明確にし、悪意のあるアカウントロックアウトを防ぐ必要がある。 | 1 |
| **6.1.2** | コンテキスト固有の単語のリストが文書化されており、パスワードでの使用を防いでいる。このリストには、組織名、製品名、システム識別子、プロジェクトコード名、部門名、ロール名などの配列を含むことがある。 | 2 |
| **6.1.3** | アプリケーションに複数の認証経路がある場合、それらすべてに一貫して適用される必要があるセキュリティ制御と認証強度がともに文書化されている。 | 2 |

## V6.2 パスワードセキュリティ

NIST SP 800-63 により「記憶された秘密 (Memorized Secrets)」と呼ばれるパスワードには、パスワード、パスフレーズ、PIN、ロック解除パターン、および正しい子猫や他の画像要素の選択があります。それらは一般に「知識認証 (something you know)」とみなされ、多くの場合に単要素認証メカニズムとして使用されます。

そのため、このセクションはパスワードが安全に作成され処理されるようにするための要件を含みます。要件のほとんどはそのレベルで最も重要であるため L1 となっています。L2 以降では、多要素認証メカニズムが必要となり、パスワードはその要素の一つとなることがあります。

このセクションの要件は主に [NIST のガイダンス](https://pages.nist.gov/800-63-3/sp800-63b.html) の [§ 5.1.1.2](https://pages.nist.gov/800-63-3/sp800-63b.html#memsecretver) に関連しています。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.2.1** | ユーザが設定するパスワードの長さは少なくとも 8 文字であるが、最低限 15 文字にすることを強く推奨している。 | 1 |
| **6.2.2** | ユーザは自身のパスワードを変更できる。 | 1 |
| **6.2.3** | パスワード変更機能にはユーザの現在のパスワードと新しいパスワードが必要とされる。 | 1 |
| **6.2.4** | アカウント登録またはパスワード変更時に送信されるパスワードは、アプリケーションのパスワードポリシー (最小長など) に合致する、少なくとも上位 3000 件の利用可能な一連のパスワードと照合されている。 | 1 |
| **6.2.5** | パスワードはどのような構成でも使用でき、許可される文字の種類を制限するルールはない。大文字、小文字、数字、特殊文字の最低数についての要求を設けてはいけない。 | 1 |
| **6.2.6** | パスワード入力フィールドは type=password を使用して、そのエントリをマスクしている。アプリケーションはユーザがマスクしたパスワード全体または最後に入力したパスワードの文字を一時的に閲覧できるようにしている。 | 1 |
| **6.2.7** | 「貼り付け」機能、ブラウザのパスワードヘルパー、および外部のパスワードマネージャが許可されている。 | 1 |
| **6.2.8** | アプリケーションは切り捨てや大文字小文字の変換などの修正をなにも加えずに、ユーザーから受け取ったパスワードを正確に検証している。 | 1 |
| **6.2.9** | 64 文字以上のパスワードが許可されている。 | 2 |
| **6.2.10** | ユーザのパスワードは侵害されたことが判明するか、ユーザーが変更するまで有効のままである。アプリケーションは定期的なクレデンシャルの変更を要求してはいけない。 | 2 |
| **6.2.11** | コンテキスト固有の単語の文書化されたリストを使用して、推測されやすいパスワードが作成されることを防いでいる。 | 2 |
| **6.2.12** | アカウント登録またはパスワード変更時に送信されるパスワードは一連の侵害されたパスワードと照合されている。 | 2 |

## V6.3 一般的な認証セキュリティ

このセクションでは、認証メカニズムのセキュリティに関する一般的な要件と、レベルごとの期待事項について説明します。L2 アプリケーションでは多要素認証 (MFA) の使用を強制する必要があります。L3 アプリケーションでは、認証済みの高信頼実行環境 (TEE) で実行される、ハードウェアベースの認証を使用する必要があります。これには、デバイスにバインドされたパスキー、eIDAS Level of Assurance (LoA) High 準拠のオーセンティケータ、NIST Authenticator Assurance Level 3 (AAL3) 保証のオーセンティケータ、または同等のメカニズムを含みます。

これは MFA に関する比較的積極的なスタンスですが、ユーザを保護するためにこれに関する基準を引き上げることが極めて重要であり、これらの要件を緩和しようとする場合には、このトピックに関する NIST のガイダンスとリサーチを考慮に入れて、認証に関するリスクをどのように軽減するかについての明確な計画を伴う必要があります。

リリース時点において、NIST SP 800-63 は電子メールを認証メカニズムとして [受け入れられない (not acceptable)](https://pages.nist.gov/800-63-FAQ/#q-b11) としていることに注意してください ([アーカイブされたコピー](https://web.archive.org/web/20250330115328/https://pages.nist.gov/800-63-FAQ/#q-b11))。

このセクションの要件は [NIST のガイダンス](https://pages.nist.gov/800-63-3/sp800-63b.html) の [§ 4.2.1](https://pages.nist.gov/800-63-3/sp800-63b.html#421-permitted-authenticator-types), [§ 4.3.1](https://pages.nist.gov/800-63-3/sp800-63b.html#431-permitted-authenticator-types), [§ 5.2.2](https://pages.nist.gov/800-63-3/sp800-63b.html#522-rate-limiting-throttling), [§ 6.1.2](https://pages.nist.gov/800-63-3/sp800-63b.html#-612-post-enrollment-binding) などのさまざまなセクションに関連しています。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.3.1** | クレデンシャルスタッフィングやパスワードブルートフォースなどの攻撃を防ぐためのコントロールは、アプリケーションのセキュリティドキュメントに従って実装されている。 | 1 |
| **6.3.2** | デフォルトユーザーアカウント ("root", "admin", "sa" など) がアプリケーションに存在しないか、無効になっている。 | 1 |
| **6.3.3** | アプリケーションにアクセスするには、多要素認証メカニズム、または単要素認証のメカニズムの組み合わせのいずれかを使用する必要がある。L3 では、ユーザによるアクション (FIDO ハードウェアキーやスマートフォンのボタン押下など) を要求することで認証の意図を検証しつつ、フィッシング攻撃に対する侵害やなりすましへの耐性を提供しているハードウェアベースの認証メカニズムが要素の一つである必要がある。この要件における考慮事項のいずれかを緩和するには、十分に文書化された根拠と包括的な緩和策を必要とする。 | 2 |
| **6.3.4** | アプリケーションに複数の認証経路がある場合、文書化されていない経路がなく、セキュリティ制御と認証強度が一貫して適用されている。 | 2 |
| **6.3.5** | 不審な認証試行 (成功または失敗) があった場合にユーザに通知されている。これには、通常とは異なる場所やクライアントからの認証試行、部分的に成功した認証 (多要素のうち一つのみ)、長期間使用されなかった後の認証試行、複数回失敗した後に成功した認証などがある。 | 3 |
| **6.3.6** | 電子メールは単要素認証メカニズムとしても多要素認証メカニズムとしても使用されない。 | 3 |
| **6.3.7** | クレデンシャルのリセットや、ユーザ名や電子メールアドレスの変更など、認証情報の更新された後に、ユーザに通知されている。 | 3 |
| **6.3.8** | エラーメッセージ、HTTP レスポンスコード、またはさまざまなレスポンスタイムなどに基づくことによって、失敗した認証チャレンジから有効なユーザーを推定することはできない。登録とパスワード忘れ機能にもこの保護が必要である。 | 3 |

## V6.4 認証要素のライフサイクルとリカバリ

認証要素にはパスワード、ソフトトークン、ハードウェアトークン、生体認証デバイスなどを含みます。これらのメカニズムのライフサイクルを安全に処理することは、アプリケーションのセキュリティにとって重要であり、このセクションはこれに関連する要件を含みます。

このセクションの要件は主に [NIST のガイダンス](https://pages.nist.gov/800-63-3/sp800-63b.html) の [§ 5.1.1.2](https://pages.nist.gov/800-63-3/sp800-63b.html#memsecretver) または [§ 6.1.2.3](https://pages.nist.gov/800-63-3/sp800-63b.html#replacement) に関連しています。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.4.1** | システムで生成される初期パスワードやアクティベーションコードは安全にランダムに生成され、既存のパスワードポリシーに従い、短期間または初期使用後に失効している。これらの初期シークレットが長期パスワードになることは許可されない。 | 1 |
| **6.4.2** | パスワードのヒントや知識ベースの認証 (いわゆる「秘密の質問」) が存在しない。 | 1 |
| **6.4.3** | 忘れたパスワードをリセットするための安全なプロセスが実装されており、有効になっている多要素認証メカニズムをバイパスしない。 | 2 |
| **6.4.4** | 多要素認証要素が失われた場合、登録時と同じレベルで同一性証明の証拠が実行される。 | 2 |
| **6.4.5** | 期限切れになる認証メカニズムの更新指示は、古い認証メカニズムの期限が切れる前に実行できるように、十分な時間をとって送信され、必要に応じて自動リマインダを設定している。 | 3 |
| **6.4.6** | 管理ユーザはユーザのパスワードリセット処理を開始できるが、ユーザのパスワードを変更したり選択することはできない。これによりユーザのパスワードを管理ユーザに知られる状況を防いでいる。 | 3 |

## V6.5 一般的な多要素認証要件

このセクションではさまざまな多要素認証方式に関連する一般的なガイダンスを提供します。

メカニズムには以下のものがあります。

* ルックアップシークレット
* 時間ベースのワンタイムパスワード (TOTP)
* 経路外メカニズム

ルックアップシークレットはトランザクション認証番号 (TAN) 、ソーシャルメディアリカバリーコードなどの、事前に生成されたシークレットコードのリスト、またはランダム値のセットを含むグリッドです。このタイプの認証メカニズムは、コードが意図的に記憶しにくいものであり、どこかに保存しておく必要があるため、「所有物認証 (something you have)」とみなされます。

時間ベースのワンタイムパスワード (Time-based, One-time Password, TOTP) は継続的に変化する疑似ランダムワンタイムチャレンジを表示する物理トークンまたはソフトトークンです。このタイプの認証メカニズムは「所有物認証 (something you have)」とみなされます。多要素トークンは単要素 TOTP と似ていますが、有効な PIN コード、生体認証ロック解除、USB 挿入または NFC ペアリングまたは追加の値 (トランザクション署名計算機など) を入力して最終ワンタイムパスワード (OTP) を作成する必要があります。

経路外メカニズムの詳細については、次のセクションで説明します。

このセクションの要件は主に [NIST のガイダンス](https://pages.nist.gov/800-63-3/sp800-63b.html) の [§ 5.1.2](https://pages.nist.gov/800-63-3/sp800-63b.html#-512-look-up-secrets), [§ 5.1.3](https://pages.nist.gov/800-63-3/sp800-63b.html#-513-out-of-band-devices), [§ 5.1.4.2](https://pages.nist.gov/800-63-3/sp800-63b.html#5142-single-factor-otp-verifiers), [§ 5.1.5.2](https://pages.nist.gov/800-63-3/sp800-63b.html#5152-multi-factor-otp-verifiers), [§ 5.2.1](https://pages.nist.gov/800-63-3/sp800-63b.html#521-physical-authenticators), [§ 5.2.3](https://pages.nist.gov/800-63-3/sp800-63b.html#523-use-of-biometrics) に関連しています。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.5.1** | ルックアップシークレット、経路外認証リクエストまたはコード、時間ベースのワンタイムパスワード (TOTP) は一度のみ正常に使用できる。 | 2 |
| **6.5.2** | アプリケーションのバックエンドに保存される際、112 ビット未満のエントロピー (19 個のランダムな英数字または 34 個のランダムな数字) を持つルックアップシークレットは、32 ビットのランダムソルトを組み込んだ承認済みのパスワードストレージハッシュアルゴリズムでハッシュされる。シークレットが 112 ビット以上のエントロピーを持つ場合は、標準的なハッシュ関数を使用できる。 | 2 |
| **6.5.3** | ルックアップシークレット、経路外認証コード、時間ベースのワンタイムパスワードのシードは、予測可能な値を避けるために暗号論的に安全な疑似乱数生成器 (Cryptographically Secure Pseudorandom Number Generator, CSPRNG) を使用して生成されている。 | 2 |
| **6.5.4** | ルックアップシークレットと経路外認証コードは最低 20 ビットのエントロピー (通常 4 個のランダムな英数字または 6 個のランダムな数字で十分です) を持っている。 | 2 |
| **6.5.5** | 経路外認証リクエスト、コード、トークン、および時間ベースのワンタイムパスワード (TOTP) は定められた有効期間を持っている。経路外リクエストでは最大有効時間を 10 分にする必要があり、TOTP では最大有効時間を 30 秒とする必要がある。 | 2 |
| **6.5.6** | あらゆる認証要素 (物理デバイスを含む) は盗難やその他の紛失の場合に無効化できる。 | 3 |
| **6.5.7** | 生体認証メカニズムは所有物認証 (something you have) または知識認証 (something you know) のいずれかと一緒に、二次的要素としてのみ使用されている。 | 3 |
| **6.5.8** | 時間ベースのワンタイムパスワード (TOTP) は、信頼できない時間やクライアントが提供する時間ではなく、信頼できるサービスからの時間ソースに基づいてチェックされている。 | 3 |

## V6.6 経路外認証メカニズム

通常、これは認証サーバが安全なセカンダリチャネルを介して物理デバイスと通信します。たとえば、モバイルデバイスへのプッシュ通知などがあります。このタイプの認証メカニズムは「所有物認証 (something you have)」とみなされます。

電子メールや VOIP などの安全でない経路外認証メカニズムは許可されていません。PSTN および SMS 認証は現在 NIST により [「制限された」認証メカニズム ("restricted" authentication mechanism)](https://pages.nist.gov/800-63-FAQ/#q-b01) と考えられており、廃止すべきであり、時間ベースのワンタイムパスワード (TOTP)、暗号メカニズム、または同様のものを優先します。NIST SP 800-63B [§ 5.1.3.3](https://pages.nist.gov/800-63-3/sp800-63b.html#-5133-authentication-using-the-public-switched-telephone-network) では、電話または SMS の経路外認証を絶対にサポートする必要がある場合、デバイス交換、SIM 変更、番号ポーティング、その他の異常な動作のリスクに対処することを推奨しています。この ASVS セクションではこれを要件として義務付けているわけではありませんが、機密性の高い L2 アプリや L3 アプリに対してこれらの予防措置を講じないことは、重大な危険信号とみなす必要があります。

また、NIST は最近 [プッシュ通知の使用を推奨しない](https://pages.nist.gov/800-63-4/sp800-63b/authenticators/#fig-3) というガイダンスも提供していることに注意してください。この ASVS セクションではプッシュ通知の使用を推奨していませんが、「プッシュ爆撃」のリスクを認識しておくことが重要です。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.6.1** | 公衆交換電話網 (PSTN) を使用して電話または SMS 経由でワンタイムパスワード (OTP) を配信する認証メカニズムは、電話番号が事前に検証され、より強力な代替手段 (時間ベースのワンタイムパスワードなど) も提供され、サービスがそのセキュリティリスクに関する情報をユーザに提供する場合にのみ提供される。L3 アプリケーションでは、電話と SMS はオプションとして利用してはいけない。 | 2 |
| **6.6.2** | 経路外認証リクエスト、コード、またはトークンはそれらが生成された元の認証リクエストにバインドされており、それ以前やそれ以降のものに対しては使用できない。 | 2 |
| **6.6.3** | コードベースの経路外認証メカニズムはレート制限を使用して、ブルートフォース攻撃から保護されている。また、少なくとも64ビットのエントロピーを持つコードの使用も検討する。 | 2 |
| **6.6.4** | プッシュ通知が多要素認証に使用される場合、レート制限を使用して、プッシュ爆撃攻撃を防いでいる。また、番号照合もこのリスクを緩和できることがある。 | 3 |

## V6.7 暗号認証メカニズム

暗号認証メカニズムはスマートカードまたは FIDO キーを含み、ユーザは認証を完了するために暗号デバイスをコンピュータに接続するかペアリングする必要があります。認証サーバはチャレンジナンスを暗号デバイスまたはソフトウェアに送信し、デバイスまたはソフトウェアは安全に保存された暗号鍵に基づいてレスポンスを計算します。このセクションの要件はこれらのメカニズムに対する実装固有のガイダンスを提供します。暗号化アルゴリズムに関するガイダンスは「暗号化」の章でカバーされています。

共有鍵や秘密鍵が暗号認証に使用される場合、「構成」の章の「シークレット管理」セクションに記載されているように、これらは他のシステムシークレットと同じメカニズムを使用して保存すべきです。

このセクションの要件は主に [NIST のガイダンス](https://pages.nist.gov/800-63-3/sp800-63b.html) の [§ 5.1.7.2](https://pages.nist.gov/800-63-3/sp800-63b.html#sfcdv) に関連しています。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.7.1** | 暗号認証アサーションを検証するために使用される証明書は、変更から保護される方法で保存されている。 | 3 |
| **6.7.2** | チャレンジナンスは少なくとも 64 ビットの長さがあり、統計的に一意か、暗号デバイスの有効期間を通じて一意となっている。 | 3 |

## V6.8 アイデンティティプロバイダによる認証

アイデンティティプロバイダ (Identity Provider, IdP) はユーザに統合されたアイデンティティを提供します。ユーザは、Azure AD、Okta、Ping Identity、Google を使用するエンタープライズアイデンティティや、Facebook、Twitter、Google、WeChat を使用するコンシューマアイデンティティなど、複数の IdP で複数のアイデンティティを持つことがよくあります。これらは一般的な代替手段のほんの一例です。このリストはこれらの企業やサービスを推奨するものではなく、多くのユーザが多くの確立されたアイデンティティを持っているという現実を考慮するよう、開発者に促すものです。組織は IdP のアイデンティティ証明の強さのリスクプロファイルに従って、既存のユーザアイデンティティとの統合を検討する必要があります。たとえば、政府機関が機密性の高いシステムのログインとしてソーシャルメディアアイデンティティを受け入れることは考えにくく、それは偽のアイデンティティを作成したり捨てることが簡単だからです。一方、モバイルゲーム会社はアクティブなプレイヤーベースを拡大するために、主要なソーシャルメディアプラットフォームと統合する必要があるかもしれません。

外部アイデンティティプロバイダを安全に使用するには、アイデンティティスプーフィングや偽造されたアサーションを防止するための慎重な構成と検証が必要です。このセクションではこれらのリスクに対処するための要件を示します。

| # | 説明 | レベル |
| --- | --- | --- |
| **6.8.1** | アプリケーションが複数のアイデンティティプロバイダ (IdP) をサポートしている場合、サポートされている別のアイデンティティプロバイダを介して (たとえば同じユーザ識別子を使用して) ユーザのアイデンティティを偽装できない。標準的な緩和策としてはアプリケーションが (名前空間として機能する) IdP ID と IdP 内のユーザ ID の組み合わせを使用して、ユーザを登録および識別することである。 | 2 |
| **6.8.2** | 認証アサーション (JWT や SAML アサーションなど) のデジタル署名の存在と完全性は常に検証され、署名されていないアサーションや無効な署名を持つアサーションは拒否される。 | 2 |
| **6.8.3** | SAML アサーションは一意に処理され、有効期間内に一度だけ使用され、リプレイ攻撃を防いでいる。 | 2 |
| **6.8.4** | アプリケーションが別のアイデンティティプロバイダ (IdP) を使用し、特定の機能に対して特定の認証強度、認証方法、認証日時を期待する場合、アプリケーションは IdP から返される情報を使用してこれを検証している。たとえば、OIDC を使用する場合、これは 'acr', 'amr', 'auth\_time' (存在する場合) などの ID トークンクレームを検証することで実現できる。IdP がこの情報を提供しない場合、アプリケーションは最低限の強度の認証メカニズム (たとえば、ユーザ名とパスワードを使用した単要素認証) が使用されたと想定する、文書化されたフォールバックアプローチを備えている必要がある。 | 2 |

## 参考情報

詳しくは以下の情報を参照してください。

* [NIST SP 800-63 - Digital Identity Guidelines](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf)
* [NIST SP 800-63B - Authentication and Lifecycle Management](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf)
* [NIST SP 800-63 FAQ](https://pages.nist.gov/800-63-FAQ/)
* [OWASP Web Security Testing Guide: Testing for Authentication](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/04-Authentication_Testing)
* [OWASP Password Storage Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html)
* [OWASP Forgot Password Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html)
* [OWASP Choosing and Using Security Questions Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Choosing_and_Using_Security_Questions_Cheat_Sheet.html)
* [CISA Guidance on "Number Matching"](https://www.cisa.gov/sites/default/files/publications/fact-sheet-implement-number-matching-in-mfa-applications-508c.pdf)
* [Details on the FIDO Alliance](https://fidoalliance.org/)

# V7 セッション管理

## 管理目標

セッション管理メカニズムは、ステートレス通信プロトコル (HTTP など) を使用している場合でも、アプリケーションがユーザとデバイスのやり取りを時間の経過とともに関連付けることができます。現代のアプリケーションは異なる特性と目的を持つ複数のセッショントークンを使用するかもしれません。安全なセッション管理システムとは、攻撃者が被害者のセッションを取得、利用、あるいはその他の方法で悪用することを防ぐものです。セッションを維持するアプリケーションは、以下のセッション管理の上位要件が満たされていることを確保する必要があります。

* セッションは各個人に固有のものであり、推測や共有することはできない。
* セッションは不要になると無効になり、非アクティブ期間内にはタイムアウトする。

この章の要件の多くは、選択された [NIST SP 800-63 Digital Identity Guidelines](https://pages.nist.gov/800-63-4/) コントロールに関連しており、一般的な脅威と一般的に悪用される認証の弱点に焦点を当てています。

特定のセッション管理メカニズムの具体的な実装の詳細に関する要件は他の場所にあることに注意してください。

* HTTP クッキーはセッショントークンを保護するための一般的なメカニズムです。クッキーの特定のセキュリティ要件は「Web フロントエンドセキュリティ」の章にあります。
* 自己完結型トークンはセッションを維持する方法として頻繁に使用されます。具体的なセキュリティ要件は「自己完結型トークン」の章にあります。

## V7.1 セッション管理ドキュメント

すべてのアプリケーションに適した単一のパターンはありません。したがって、すべてのケースに適した普遍的な境界と制限を定義することは現実的ではありません。実装とテストの前提条件として、セッション処理に関連するセキュリティ上の決定を文書化したリスク分析を実施する必要があります。これにより、セッション管理システムがアプリケーションの特定の要件に適合するようになります。

ステートフルと「ステートレス」のどちらのセッションメカニズムを選択するかに関わらず、選択されたソリューションが関連するすべてのセキュリティ要件を満たすことができることを示すために、分析を完了して文書化する必要があります。使用中のシングルサインオン (SSO) メカニズムとのやり取りも考慮する必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **7.1.1** | ユーザのセッション非アクティブタイムアウトと絶対最大セッション有効期間が文書化され、他のコントロールと組み合わせて適切であり、そのドキュメントには NIST SP 800-63B 再認証要件からの逸脱の正当性が含まれている。 | 2 |
| **7.1.2** | ドキュメントには、一つのアカウントで許可される同時 (並列) セッションの数と、アクティブセッションの最大数に達した場合に実行される意図した動作やアクションを定義している。 | 2 |
| **7.1.3** | フェデレーション ID (federated identity) 管理エコシステムの一部としてユーザセッションを作成および管理するすべてのシステム (SSO システムなど) は、セッションの有効期間、終了、および再認証を必要とするその他の状態を調整するためのコントロールとともに文書化している。 | 2 |

## V7.2 基本セッション管理セキュリティ

このセクションでは、セッショントークンが安全に生成および検証されることを確認することで、安全なセッションの基本的な要件を満たします。

| # | 説明 | レベル |
| --- | --- | --- |
| **7.2.1** | アプリケーションが信頼できるバックエンドサービスを使用してすべてのセッショントークン検証を実行している。 | 1 |
| **7.2.2** | アプリケーションがセッション管理に動的に生成される自己完結型トークンまたはリファレンストークンを使用している。つまり、静的な API シークレットと API キーは使用していない。 | 1 |
| **7.2.3** | リファレンストークンがユーザセッションを表すために使用される場合、そのトークンは一意であり、暗号論的に安全な疑似乱数生成器 (CSPRNG) を使用して生成され、少なくとも 128 ビットのエントロピーを持つ。 | 1 |
| **7.2.4** | アプリケーションがユーザ認証 (再認証を含む) 時に新しいセッショントークンを生成し、現在のセッショントークンを終了する。 | 1 |

## V7.3 セッションタイムアウト

セッションタイムアウトメカニズムは、セッションハイジャックやその他の形式のセッション悪用の機会を最小限に抑えるのに役立ちます。タイムアウトは文書化されたセキュリティ上の決定を満たす必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **7.3.1** | 文書化されたセキュリティ上の決定に従って再認証が強制されるような、非アクティブタイムアウトがある。 | 2 |
| **7.3.2** | リスク分析と文書化されたセキュリティ上の決定に従って再認証が強制されるような、絶対的な最大セッション存続期間がある。 | 2 |

## V7.4 セッションの終了

セッションの終了はアプリケーション自体によって処理されるか、SSO プロバイダ がアプリケーションの代わりにセッション管理を処理している場合は SSO プロバイダによって処理されます。いくつかの要件はプロバイダによってコントロールされる可能性があるため、このセクションの要件を検討する際に、SSO プロバイダが適用範囲に含まれるかどうかを判断する必要があるかもしれません。

セッションの終了は再認証を必要とし、アプリケーション、フェデレーションログイン (存在する場合)、依拠当事者 (Relying Party) すべてに有効となっている必要があります。

ステートフルセッションメカニズムでは、終了には一般的にバックエンドでセッションの無効にすることを含みます。自己完結型トークンの場合、トークンが有効期限まで有効なままになる可能性があるため、トークンを失効するかブロックするための追加の対策が必要です。

| # | 説明 | レベル |
| --- | --- | --- |
| **7.4.1** | セッションの終了 (ログアウトや期限切れなど) がトリガーされると、アプリケーションはセッションのそれ以上の使用を禁止している。リファレンストークンやステートフルセッションでは、これはアプリケーションバックエンドでセッションデータを無効にすることを意味する。自己完結型トークンを使用するアプリケーションでは、終了したトークンのリストを保持する、ユーザごとに所定の日時より前に生成されたトークンを禁止する、ユーザごとに署名鍵を入れ替えるなどの解決策が必要になる。 | 1 |
| **7.4.2** | ユーザアカウントが無効または削除された場合 (従業員の退職など) 、アプリケーションはすべてのアクティブなセッションを終了する。 | 1 |
| **7.4.3** | 認証要素が正常に変更または削除 (リセットやリカバリーによるパスワード変更や、存在する場合、MFA 設定の更新を含む) された後、アプリケーションが他のすべてのアクティブセッションを終了するためのオプションを提供している。 | 2 |
| **7.4.4** | 認証を必要とするすべてのページでログアウト機能に簡単かつ目に見える形でアクセスできる。 | 2 |
| **7.4.5** | アプリケーション管理者は個々のユーザーまたはすべてのユーザーのアクティブなセッションを終了できる。 | 2 |

## V7.5 セッションの悪用に対する防御

このセクションでは、アクティブユーザセッションの存在と機能に依存するベクトルを通じて、ハイジャックや悪用されるアクティブセッションによってもたらされるリスクを軽減するための要件を提供します。たとえば、悪意のあるコンテンツ実行を使用して、認証された被害者のブラウザに被害者のセッションを使用してアクションを実行します。

このセクションの要件を検討する際には、「認証」の章のレベル固有のガイダンスを考慮する必要があることに注意してください。

| # | 説明 | レベル |
| --- | --- | --- |
| **7.5.1** | アプリケーションは、電子メールアドレス、電話番号、MFA 構成、アカウントリカバリに使用されるその他の情報など、認証に影響する可能性のある機密性の高いアカウント属性の変更を許可する前に、完全な再認証を要求している。 | 2 |
| **7.5.2** | ユーザは、現在のアクティブセッションの一部またはすべてを表示し、(少なくとも一つの要素で再度認証することで) 終了することができる。 | 2 |
| **7.5.3** | アプリケーションは、機密性の高いトランザクションや操作を実行する前に、少なくとも一つの要素での追加の認証または二次検証を要求している。 | 3 |

## V7.6 フェデレーション再認証

このセクションは依拠当事者 (Relying Party, RP) やアイデンティティプロバイダ (Identity Provider, IdP) コードを作成する人に関連します。これらの要件はフェデレーションおよびアサーションに関する [NIST SP 800-63C](https://pages.nist.gov/800-63-4/sp800-63c.html) に由来します。

| # | 説明 | レベル |
| --- | --- | --- |
| **7.6.1** | 依拠当事者 (Relying Party, RP) とアイデンティティプロバイダ (Identity Provider, IdP) 間のセッションの有効期間と終了は文書化されたとおりに動作し、IdP 認証イベント間の最大時間に達した場合など、必要に応じて再認証を要求している。 | 2 |
| **7.6.2** | セッションの作成にはユーザの同意または明示的なアクションが必要であり、ユーザとのやり取りなしに新しいアプリケーションセッションが作成されることを防いでいる。 | 2 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP Web Security Testing Guide: Session Management Testing](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/06-Session_Management_Testing)
* [OWASP Session Management Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html)

# V8 認可

## 管理目標

認可は、許可されたコンシューマ (ユーザ、サーバ、その他のクライアント) にのみアクセスを付与することを確保します。最小権限の原則 (POLP) を適用するために、検証対象のアプリケーションが以下の上位要件を満たす必要があります。

* 意思決定要因や環境コンテキストを含む認可ルールを文書化している。
* コンシューマは定義された権限によって許可されたリソースにのみアクセスできる必要がある。

## V8.1 認可ドキュメント

包括的な認可ドキュメントは、セキュリティ上の決定が一貫して適用され、監査可能であり、組織のポリシーに準拠していることを確保するために不可欠です。これにより、開発者、管理者、テスト担当者にとってセキュリティ要件が明確で実行可能なものとなり、認可されていないアクセスのリスクを軽減します。

| # | 説明 | レベル |
| --- | --- | --- |
| **8.1.1** | 認可ドキュメントは、コンシューマのパーミッションとリソース属性に基づいて、機能レベルとデータ固有のアクセスを制限するためのルールを定義している。 | 1 |
| **8.1.2** | 認可ドキュメントは、コンシューマのパーミッションとリソース属性に基づいて、フィールドレベルのアクセス (読み取りと書き込みの両方) を制限するためのルールを定義している。これらのルールは、状態やステータスなど、関連するデータオブジェクトの他の属性値に依存する可能性があることに注意する。 | 2 |
| **8.1.3** | アプリケーションのドキュメントは、認証と認可に関連するものを含め、セキュリティ上の決定をするために、アプリケーションで使用される環境属性とコンテキスト属性 (時間帯、ユーザの位置情報、IP アドレス、デバイスなどを含むがこれらに限定されない) を定義している。 | 3 |
| **8.1.4** | 認証と認可のドキュメントは、機能レベル、データ固有、フィールドレベルの認可に加えて、環境要因とコンテキスト要因が意思決定でどのように使用されるかを定義している。これは、評価される属性、リスクの閾値、実行されるアクション (許可、チャレンジ、拒否、ステップアップ認証など) を含む必要がある。 | 3 |

## V8.2 一般的な認可設計

機能、データ、フィールドレベルできめ細かい認可コントロールを実装することで、コンシューマは明示的に許可されたものだけにアクセスできるようになります。

| # | 説明 | レベル |
| --- | --- | --- |
| **8.2.1** | アプリケーションは機能レベルのアクセスが明示的なパーミッションを持つコンシューマに制限されることを確保している。 | 1 |
| **8.2.2** | アプリケーションは、安全でない直接オブジェクト参照 (IDOR) と壊れたオブジェクトレベル認可 (BOLA) を軽減するために、データ固有のアクセスが特定のデータ項目に対する明示的なパーミッションを持つコンシューマに制限されることを確保している。 | 1 |
| **8.2.3** | アプリケーションは、壊れたオブジェクトプロパティレベル認可 (BOPLA) を軽減するために、フィールドレベルのアクセスが特定のフィールドに対する明示的なパーミッションを持つコンシューマに制限されることを確保している。 | 2 |
| **8.2.4** | アプリケーションのドキュメントに定義されているように、認証と認可の決定には、コンシューマの環境属性とコンテキスト属性 (時間帯、位置情報、IP アドレス、デバイスなど) に基づく適応型セキュリティコントロールが実装されている。これらのコントロールは、コンシューマが新しいセッションを開始しようとするときだけでなく、既存のセッション中にも適用する必要がある。 | 3 |

## V8.3 操作レベルの認可

特に動的な環境では、認可されていないアクションを防ぐために、アプリケーションのアーキテクチャの適切な層で認可の変更を即座に適用することが重要です。

| # | 説明 | レベル |
| --- | --- | --- |
| **8.3.1** | アプリケーションは信頼できるサービス層で認可ルールを適用し、クライアントサイド JavaScript など、信頼できないコンシューマが操作できるコントロールに依存していない。 | 1 |
| **8.3.2** | 認可の決定が行われた値の変更は即座に適用されている。変更を即座に適用できない場合 (自己完結型トークンのデータに依存している場合など)、コンシューマがもはや認可されていないアクションを実行したときに警告を発し、その変更を元に戻すための緩和コントロールが必要である。この代替手段では情報漏洩を緩和しないことに注意する。 | 3 |
| **8.3.3** | オブジェクトへのアクセスは、発信主体 (コンシューマなど) のパーミッションに基づいており、代理で動作する仲介者やサービスのパーミッションではない。たとえば、コンシューマが認証のために自己完結型トークンを使用して Web サービスを呼び出し、それからそのサービスが別のサービスにデータをリクエストする場合、二番目のサービスは、最初のサービスからのマシン間トークンではなく、コンシューマのトークンを使用してパーミッションを決定する。 | 3 |

## V8.4 他の認可の考慮

特に管理インタフェースやマルチテナント環境では、認可をさらに考慮することで、認可されていないアクセスの防止に役立ちます。

| # | 説明 | レベル |
| --- | --- | --- |
| **8.4.1** | マルチテナントアプリケーションはクロステナントコントロールを使用して、コンシューマ操作が、インタラクションするパーミッションを持たないテナントに決して影響を与えないようにしている。 | 2 |
| **8.4.2** | 管理インタフェースへのアクセスは、継続的なコンシューマアイデンティティ検証、デバイスセキュリティ態勢評価、コンテキストリスク分析など、複数レイヤのセキュリティを組み込んでおり、ネットワークの場所や信頼できるエンドポイントが、認可されていないアクセスの可能性を減らすことはあるものの、認可の唯一の要素にはならないようにしている。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP Web Security Testing Guide: Authorization](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/05-Authorization_Testing)
* [OWASP Authorization Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html)

# V9 自己完結型トークン

## 管理目標

自己完結型トークンという概念は 2012 年のオリジナルの RFC 6749 OAuth 2.0 で言及されています。これは受信側サービスがセキュリティ上の決定を行うために依拠するデータやクレームを含むトークンを指します。これは、受信側サービスがローカルでデータを検索するために使用する識別子のみを含む単純なトークンとは区別されます。自己完結型トークンの最も一般的な例は JSON Web Token (JWT) と SAML アサーションです。

自己完結型トークンの使用は、OAuth と OIDC 以外でも非常に広まっています。同時に、このメカニズムのセキュリティは、トークンの完全性を検証し、トークンが特定のコンテキストに対して有効であることを確保する機能に依存しています。このプロセスには多くの落とし穴があり、この章ではそれらを防ぐためにアプリケーションが備えておくべきメカニズムの詳細を具体的に説明します。

## V9.1 トークンのソースと完全性

このセクションは、トークンが信頼できるパーティによって生成され、改竄されていないことを確認するための要件を含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **9.1.1** | 自己完結型トークンがデジタル署名または MAC を使用して検証され、トークンのコンテンツを受け入れる前に改竄を防いでいる。 | 1 |
| **9.1.2** | 特定のコンテキストでは、許可リストにあるアルゴリズムのみを使用して、自己完結型トークンを作成および検証している。許可リストには、許可されたアルゴリズム、理想的には対称アルゴリズムあるいは非対称アルゴリズムのいずれかのみ、を含める必要があり、'None' アルゴリズムを含めてはいけない。対称と非対称の両方がサポートされる必要がある場合は、追加のコントロールで鍵の混乱を防ぐ必要がある。 | 1 |
| **9.1.3** | 自己完結型トークンを検証するために使用される鍵マテリアルは、トークン発行者の事前設定済みの信頼できるソースからのものであり、攻撃者が信頼できないソースや鍵を指定することを防いでいる。JWT やその他の JWS 構造では、'jku', 'x5u', 'jwk' などのヘッダは信頼できるソースの許可リストに照らして検証しなけれならない。 | 1 |

## V9.2 トークンコンテンツ

自己完結型トークンのコンテンツに基づいてセキュリティ上の決定を行う前に、そのトークンが有効期間内に提示されたこと、および提示された目的のために受信側サービスによって使用されることを検証する必要があります。これは、異なるサービス間や同じ発行者からの異なるトークンタイプでの安全でない相互使用を避けるのに役立ちます。

OAuth と OIDC の具体的な要件については専用の章で説明します。

| # | 説明 | レベル |
| --- | --- | --- |
| **9.2.1** | トークンデータに有効期間がある場合、トークンとそのコンテンツは検証時間がこの有効期間内である場合にのみ受け入れられる。たとえば、JWT では、クレーム 'nbf' と 'exp' を検証する必要がある。 | 1 |
| **9.2.2** | トークンを受け取るサービスは、トークンの内容を受け入れる前に、トークンが正しいタイプであり、意図した目的に適していることを検証している。たとえば、認可の決定にはアクセストークンのみを受け入れることができ、ユーザ認証の証明には ID トークンのみを使用できる。 | 2 |
| **9.2.3** | サービスはそのサービス (オーディエンス) で使用することを意図したトークンのみを受け入れている。JWT では、これはサービス内で定義された許可リストに対して 'aud' クレーム を検証することで達成できる。 | 2 |
| **9.2.4** | トークン発行者が同じ秘密鍵を使用して異なるオーディエンスにトークンを発行する場合、発行されるトークンには意図したオーディエンスを一意に識別するオーディエンス制限を含む。これにより、トークンが意図しないオーディエンスに再使用されることを防止している。オーディエンス識別子は動的にプロビジョンされる場合、トークン発行者はこれらのオーディエンスを検証して、オーディエンスのなりすましが発生しないようにする必要がある。 | 2 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP JSON Web Token Cheat Sheet for Java Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html) (なお役に立つ一般的なガイダンスがあります)

# V10 OAuth と OIDC

## 管理目標

OAuth2 (この章では OAuth と呼びます) は委譲された認可のための業界標準フレームワークです。たとえば、OAuth を使用すると、ユーザがクライアントアプリケーションに認可を与えている場合、クライアントアプリケーションはユーザに代わって API (サーバリソース) へのアクセスを取得できます。

それ自体では、OAuth はユーザ認証用に設計されていません。OpenID Connect (OIDC) フレームワークは、OAuth の上にユーザアイデンティティレイヤを追加することで OAuth を拡張します。OIDC は、標準化されたユーザ情報、シングルサインオン (SSO)、セッション管理などの機能をサポートします。OIDC は OAuth の拡張であるため、この章の OAuth 要件は OIDC にも適用します。

OAuth では以下のロールが定義されています。

* OAuth クライアントは、サーバリソースへのアクセスを (発行されたアクセストークンを使用して API を呼び出すなどで) 取得しようとするアプリケーションです。OAuth クライアントは、多くの場合、サーバサイドアプリケーションです。
  + 機密クライアントは、認可サーバで自身を認証するために使用するクレデンシャルの機密性を維持できるクライアントです。
  + パブリッククライアントは、認可サーバで自身を認証するためのクレデンシャルの機密性を維持できません。したがって、自身を認証 ('client\_id' と 'client\_secret' パラメータを使用するなど) する代わりに、自身を識別 ('client\_id' パラメータを使用する) します。
* OAuth リソースサーバ (RS) は OAuth クライアントにリソースを公開するサーバ API です。
* OAuth 認可サーバ (AS) は OAuth クライアントにアクセストークンを発行するサーバアプリケーションです。これらのアクセストークンにより、OAuth クライアントはエンドユーザに代わって、あるいは OAuth クライアント自身の代わりとして RS リソースにアクセスできます。AS は独立したアプリケーションであることが多いのですが、(適切な場合) 適切な RS に統合されることもあります。
* リソースオーナー (RO) は、OAuth クライアントがリソースサーバでホストされているリソースへの限定的なアクセスを取得することを自分の代わりに認可するエンドユーザです。リソースオーナーは認可サーバとやり取りすることでこの委譲された認可に同意します。

OIDC では以下のロールが定義されています。

* 依拠当事者 (RP) は OpenID プロバイダを通じてエンドユーザ認証を要求するクライアントアプリケーションです。OAuth クライアントのロールを担います。
* OpenID プロバイダ (OP) は、エンドユーザを認証し、OIDC クレームを RP に提供できる OAuth AS です。OP はアイデンティティプロバイダ (IdP) であることもありますが、連合シナリオでは、OP とアイデンティティプロバイダ (エンドユーザが認証する場所) は異なるサーバアプリケーションであることもあります。

OAuth と OIDC は当初はサードパーティアプリケーション向けに設計されていました。現在は、ファーストパーティアプリケーションでも使用されることがよくあります。しかし、認証やセッション管理など、ファーストパーティシナリオで使用する場合、プロトコルは複雑さを増し、新たなセキュリティ上の課題を引き起こす可能性があります。

OAuth と OIDC は多くの種類のアプリケーションに使用できますが、ASVS とこの章の要件の焦点は Web アプリケーションと API です。

OAuth と OIDC は Web テクノロジの上のロジックとみなすことができるため、他の章にある一般的な要件を常に適用します。この章をコンテキストから外すことはできません。

この章では <https://oauth.net/2/> と <https://openid.net/developers/specs/> にある仕様に沿った OAuth2 と OIDC の現在のベストプラクティスを説明します。RFC は成熟していると考えられていますが、頻繁に更新されます。したがって、この章の要件を適用する際には、最新バージョンに合わせることが重要です。詳細については参考情報セクションを参照してください。

この領域の複雑さを考えると、安全な OAuth または OIDC ソリューションには、よく知られている業界標準の認可サーバを使用し、推奨されるセキュリティ構成を適用することが極めて重要です。

この章で使用される用語は OAuth RFC および OIDC 仕様に準拠していますが、OIDC 用語は OIDC 固有の要件にのみ使用され、それ以外は OAuth 用語が使用されることに注意してください。

OAuth および OIDC のコンテキストでは、「トークン」という用語は以下のものを指します。

* アクセストークン。これは RS によってのみ使用されるものとし、イントロスペクションを使用して検証されるリファレンストークン、または何らかの鍵マテリアルを使用して検証される自己完結型トークンのいずれかになります。
* リフレッシュトークン。これはトークンを発行した認可サーバによってのみ使用されます。
* OIDC ID トークン。これは認可フローをトリガーしたクライアントによってのみ使用されます。

この章のいくつかの要件のリスクレベルは、クライアントが機密クライアントであるか、パブリッククライアントとみなされるかによって異なります。強力なクライアント認証を使用することで多くの攻撃ベクトルを軽減するため、L1 アプリケーションに機密クライアントを使用する場合、いくつかの要件が緩和される可能性があります。

## V10.1 一般的な OAuth と OIDC セキュリティ

このセクションは OAuth や OIDC を使用するすべてのアプリケーションに適用する一般的なアーキテクチャ要件をカバーします。

| # | 説明 | レベル |
| --- | --- | --- |
| **10.1.1** | トークンは厳密に必要とするコンポーネントにのみ送信される。たとえば、ブラウザベースの JavaScript アプリケーションで Backend for Frontend パターンを使用する場合、アクセストークンとリフレッシュトークンはバックエンドでのみアクセス可能である。 | 2 |
| **10.1.2** | クライアントは、認可サーバからの値 (認可コードや ID トークンなど) が、同じユーザエージェントセッションとトランザクションによって開始された認可フローから得られた値である場合にのみ、その値を受け入れている。これには、コード交換のための証明鍵 (PKCE) 'code\_verifier'、'state'、OIDC 'nonce' など、クライアントが生成したシークレットが推測不可能であり、トランザクションに固有であり、トランザクションが開始されたクライアントとユーザエージェントセッションの両方に安全にバインドされていることが必要である。 | 2 |

## V10.2 OAuth クライアント

これらの要件は OAuth クライアントアプリケーションの責務を詳述しています。クライアントには、たとえば、Web サーババックエンド (多くの場合 Backend For Frontend (BFF) として機能します)、バックエンドサービス統合、フロントエンドシングルページアプリケーション (SPA、別名ブラウザベースのアプリケーション) があります。

一般的に、バックエンドクライアントは機密クライアントとみなされ、フロントエンドクライアントはパブリッククライアントとみなされます。しかし、OAuth 動的クライアント登録を実行する場合、エンドユーザデバイスで実行するネイティブアプリケーションは機密クライアントとみなされる可能性があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **10.2.1** | コードフローが使用される場合、OAuth クライアントはコード交換のための証明鍵 (PKCE) 機能を使用するか、認可リクエストで送信された state パラメータをチェックすることで、トークンリクエストをトリガーする、一般にクロスサイトリクエストフォージェリ (CSRF) として知られるブラウザベースのリクエストフォージェリ攻撃から保護している。 | 2 |
| **10.2.2** | OAuth クライアントが複数の認可サーバとやり取りできる場合、ミックスアップ攻撃に対して防御している。たとえば、認可サーバが 'iss' パラメータ値を返し、認可レスポンスとトークンレスポンスでそれを検証することを要求できる。 | 2 |
| **10.2.3** | OAuth クライアントは、認可サーバへのリクエストで必要なスコープ (またはその他の認可パラメータ) のみをリクエストしている。 | 3 |

## V10.3 OAuth リソースサーバ

ASVS とこの章のコンテキストでは、リソースサーバは API です。安全なアクセスを提供するには、リソースサーバは以下を行う必要があります。

* トークン形式と関連するプロトコル仕様 (JWT バリデーションや OAuth トークンイントロスペクションなど) に従って、アクセストークンを検証します。
* 有効な場合、アクセストークンの情報と付与されたパーミッションに基づいて認可決定を実施します。たとえば、リソースサーバは、クライアント (RO に代わって動作する) が要求されたリソースにアクセスするための認可があることを検証する必要があります。

したがって、ここでリストされている要件は OAuth や OIDC に固有のものであり、トークンバリデーション後、トークンの情報に基づいて認可を実行する前に実行する必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **10.3.1** | リソースサーバは、そのサービス (オーディエンス) での使用を意図したアクセストークンのみを受け付けている。オーディエンスは、構造化されたアクセストークン (JWT の 'aud' クレームなど) に含めることや、トークンイントロスペクションエンドポイントを使用してチェックできる。 | 2 |
| **10.3.2** | リソースサーバは、デリゲートされた認可を定義するアクセストークンからのクレームに基づいて認可決定を実施している。'sub', 'scope', 'authorization\_details' などのクレームが存在する存在する場合、それらは決定の一部となる必要がある。 | 2 |
| **10.3.3** | アクセス制御の決定がアクセストークン (JWT または関連するトークンイントロスペクションレスポンス) から一意のユーザを識別する必要がある場合、リソースサーバは他のユーザーに再割り当てできないクレームからユーザを識別する。通常、これは 'iss' と 'sub' のクレームの組み合わせを使用することを意味する。 | 2 |
| **10.3.4** | リソースサーバが特定の認証強度、認証方法、認証日時を要求する場合、リソースサーバは提示されたアクセストークンがこれらの制約を満たすことを検証している。たとえば、存在する場合、それぞれ OIDC 'acr', 'amr', 'auth\_time' クレームを使用する。 | 2 |
| **10.3.5** | リソースサーバは、OAuth 2 の相互 TLS または OAuth 2 Demonstration of Proof of Possession (DPoP) のいずれかの送信者制約のあるアクセストークンを要求することで、盗まれたアクセストークンの使用やアクセストークンのリプレイを (認可されていない第三者から) 防止する。 | 3 |

## V10.4 OAuth 認可サーバ

これらの要件は OpenID プロバイダを含む OAuth 認可サーバの責務を詳述しています。

クライアント認証では、[RFC 8705](https://datatracker.ietf.org/doc/html/rfc8705) の [セクション 2.2](https://datatracker.ietf.org/doc/html/rfc8705#name-self-signed-certificate-mut) で要求される前提条件を満たした 'self\_signed\_tls\_client\_auth' メソッドが許可されています。

| # | 説明 | レベル |
| --- | --- | --- |
| **10.4.1** | 認可サーバは、正確な文字列比較を使用して、事前登録された URI のクライアント固有の許可リストに基づいて、リダイレクト URI を検証している。 | 1 |
| **10.4.2** | 認可サーバが認可レスポンスで認可コードを返す場合、そのコードはトークンリクエストに対して一度のみ使用できる。アクセストークンの発行にすでに使用されている認可コードでの二回目の有効なリクエストに対しては、認可サーバはトークンリクエストを拒否して、認可コードに関連する発行済みトークンをすべて取り消す必要がある。 | 1 |
| **10.4.3** | 認可コードは有効期間を短くしている。最大有効期間は、L1 および L2 アプリケーションで 10 分まで、L3 アプリケーションで 1 分までである。 | 1 |
| **10.4.4** | 特定のクライアントに対して、認可サーバはこのクライアントが使用する必要があるグラントの使用のみを許可している。'token' (暗黙的フロー) と 'password' (リソースオーナーのパスワードクレデンシャルフロー) のグラントはもはや使用してはならないことに注意する。 | 1 |
| **10.4.5** | 認可サーバは、できれば送信者制約リフレッシュトークン、つまり、Demonstrating Proof of Possession (DPoP)、または相互 TLS (mTLS) を使用した Certificate-Bound Access Tokens を使用して、パブリッククライアントに対するリフレッシュトークンリプレイ攻撃を軽減している。L1 および L2 アプリケーションの場合、リフレッシュトークンローテーションを使用できる。リフレッシュトークンローテーションを使用する場合、認可サーバが使用後にリフレッシュトークンを無効化する必要があり、すでに使用されて無効化されたリフレッシュトークンが提供された場合には、その認可のためのすべてのリフレッシュトークンを失効している。 | 1 |
| **10.4.6** | コードグラントを使用する場合、認可サーバはコード交換のための証明鍵 (PKCE) を要求することで認可コード傍受攻撃を軽減している。認可リクエストでは、認可サーバは有効な 'code\_challenge' 値を要求する必要があり、'plain' の 'code\_challenge\_method' 値を受け入れてはいけない。トークンリクエストでは、'code\_verifier' パラメータのバリデーションを要求する必要がある。 | 2 |
| **10.4.7** | 認可サーバが認可されていない動的クライアント登録をサポートしている場合、悪意のあるクライアントアプリケーションのリスクを緩和している。登録された URI などのクライアントメタデータを検証し、ユーザーの同意を確保し、信頼できないクライアントアプリケーションでの認可リクエストを処理する前にユーザに警告する必要がある。 | 2 |
| **10.4.8** | リフレッシュトークンは、スライディングリフレッシュトークン有効期限が適用されている場合も含め、絶対的な有効期限を持つ。 | 2 |
| **10.4.9** | 悪意のあるクライアントや盗まれたトークンのリスクを軽減するために、認可されたユーザが認可サーバのユーザインタフェースを使用して、リフレッシュトークンとリファレンスアクセストークンを失効させることができる。 | 2 |
| **10.4.10** | コンフィデンシャルクライアントは、トークンリクエスト、プッシュ認可リクエスト (PAR)、トークン失効リクエストなどのクライアントから認可済みサーバへのバックチャネルリクエストに対して認証されている。 | 2 |
| **10.4.11** | 認可サーバ設定では、必要なスコープのみを OAuth クライアントに割り当てている。 | 2 |
| **10.4.12** | 特定のクライアントに対して、認可サーバはこのクライアントが使用する必要がある 'response\_mode' 値のみを許可している。たとえば、認可サーバがこの値を期待値と検証したり、プッシュ認可リクエスト (PAR) や JWT で保護された認可リクエスト (JAR) を使用している。 | 3 |
| **10.4.13** | 'code' グラントタイプは常にプッシュ認可リクエスト (PAR) と一緒に使用されている。 | 3 |
| **10.4.14** | 認可サーバは、相互 TLS (mTLS) を使用した証明書バインドアクセストークン、または DPoP バインドアクセストークン (Demonstration of Proof of Possession) のいずれかを使用して、送信者制約 (Proof-of-Possession) アクセストークンのみを発行する。 | 3 |
| **10.4.15** | サーバサイドクライアント (エンドユーザデバイス上で実行されない) では、認可サーバは 'authorization\_details' パラメータ値がクライアントバックエンドからのものであり、ユーザがそれを改竄していないことを確保している。たとえば、プッシュ認可リクエスト (PAR) または JWT で保護された認可リクエスト (JAR) の使用を要求する。 | 3 |
| **10.4.16** | クライアントはコンフィデンシャルであり、認可サーバは相互 TLS ('tls\_client\_auth', 'self\_signed\_tls\_client\_auth') や秘密鍵 JWT ('private-key-jwt') などの強力なクライアント認証方法 (公開鍵暗号に基づき、リプレイ攻撃に耐性がある) の使用を要求する。 | 3 |

## V10.5 OIDC クライアント

OIDC 依拠当事者は OAuth クライアントとして機能するため、「OAuth クライアント」セクションの要件も適用します。

「認証」の章の「アイデンティティプロバイダによる認証」セクションにも、関連する一般的な要件を含むことに注意してください。

| # | 説明 | レベル |
| --- | --- | --- |
| **10.5.1** | クライアント (依拠当事者) は ID トークンのリプレイ攻撃を軽減している。たとえば、ID トークンの 'nonce' クレームが OpenID プロバイダへの認証リクエスト (OAuth2 では認可サーバに送信される認可リクエストと呼ばれる) で送信された 'nonce' 値と一致することを確認している。 | 2 |
| **10.5.2** | クライアントは ID トークンクレーム (通常は 'sub' クレーム) からユーザを一意に識別するが、(ID プロバイダのスコープでは) 他のユーザに再割り当てできない。 | 2 |
| **10.5.3** | クライアントは悪意のある認可サーバが認可サーバメタデータを通じて別の認可サーバになりすまそうとする試みを拒否する。認可サーバメタデータの発行者 URL が、クライアントが期待する事前設定された発行者 URL と正確に一致しない場合、クライアントは認可サーバメタデータを拒否する必要がある。 | 2 |
| **10.5.4** | クライアントは、トークンの 'aud' クレームがクライアントの 'client\_id' 値と等しいことをチェックすることで、ID トークンがそのクライアント (オーディエンス) のために使用されることを意図していることを検証している。 | 2 |
| **10.5.5** | OIDC バックチャネルログアウトを使用する場合、依拠当事者は強制ログアウトと、ログアウトフローにおける JWT 間の混乱によってサービス拒否を緩和している。クライアントは、ログアウトトークンが 'logout+jwt' の値で正しくタイプされていること、正しいメンバー名を持つ 'event' クレームを含むこと、'nonce' クレームを含まないことを検証する必要がある。有効期限を短く (例: 2 分) 設定することも推奨されることに注意する。 | 2 |

## V10.6 OpenID プロバイダ

OpenID プロバイダは OAuth 認可サーバとして機能するため、「OAuth 認可サーバ」セクションの要件も適用します。

ID トークンフロー (コードフローではない) を使用する場合、アクセストークンは発行されず、OAuth AS の要件の多くは適用されないことに注意してください。

| # | 説明 | レベル |
| --- | --- | --- |
| **10.6.1** | OpenID プロバイダはレスポンスモードに 'code', 'ciba', 'id\_token', 'id\_token code' の値のみを許可している。'code' は 'id\_token code' (OIDC ハイブリッドフロー) よりも優先されること、'token' (任意の暗黙的フロー) は使用してはならないことに注意する。 | 2 |
| **10.6.2** | OpenID プロバイダは強制ログアウトによってサービス拒否を緩和している。エンドユーザから明示的な確認を得るか、'id\_token\_hint' などの (依拠当事者によって開始される) ログアウトリクエスト内のパラメータが存在する場合はそれを検証している。 | 2 |

## V10.7 同意管理

これらの要件は認可サーバによるユーザの同意の検証をカバーします。適切なユーザ同意の検証を行わないと、悪意にある行為者がなりすましやソーシャルエンジニアリングを通じてユーザに代わってパーミッションを取得する可能性があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **10.7.1** | 認可サーバは、ユーザが各認可リクエストに同意することを確保している。クライアントのアイデンティティを保証できない場合、認可サーバは常に明示的にユーザに同意を求める必要がある。 | 2 |
| **10.7.2** | 認可サーバはユーザの同意を求める場合、同意される内容について十分かつ明確な情報を提示している。該当する場合、これにはリクエストされた認可の性質 (通常はスコープ、リソースサーバ、リッチ認可リクエスト (RAR) 認可の詳細に基づく)、認可されたアプリケーションのアイデンティティ、これらの認可の有効期間を含む。 | 2 |
| **10.7.3** | ユーザは、認可サーバを通じてユーザが付与した同意を確認、変更、取消できる。 | 2 |

## 参考情報

OAuth について詳しくは以下の情報を参照してください。

* [oauth.net](https://oauth.net/)
* [OWASP OAuth 2.0 Protocol Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/OAuth2_Cheat_Sheet.html)

ASVS の OAuth 関連の要件については、以下の発行済みおよびドラフト状態の RFC が使用されます。

* [RFC6749 The OAuth 2.0 Authorization Framework](https://datatracker.ietf.org/doc/html/rfc6749)
* [RFC6750 The OAuth 2.0 Authorization Framework: Bearer Token Usage](https://datatracker.ietf.org/doc/html/rfc6750)
* [RFC6819 OAuth 2.0 Threat Model and Security Considerations](https://datatracker.ietf.org/doc/html/rfc6819)
* [RFC7636 Proof Key for Code Exchange by OAuth Public Clients](https://datatracker.ietf.org/doc/html/rfc7636)
* [RFC7591 OAuth 2.0 Dynamic Client Registration Protocol](https://datatracker.ietf.org/doc/html/rfc7591)
* [RFC8628 OAuth 2.0 Device Authorization Grant](https://datatracker.ietf.org/doc/html/rfc8628)
* [RFC8707 Resource Indicators for OAuth 2.0](https://datatracker.ietf.org/doc/html/rfc8707)
* [RFC9068 JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens](https://datatracker.ietf.org/doc/html/rfc9068)
* [RFC9126 OAuth 2.0 Pushed Authorization Requests](https://datatracker.ietf.org/doc/html/rfc9126)
* [RFC9207 OAuth 2.0 Authorization Server Issuer Identification](https://datatracker.ietf.org/doc/html/rfc9207)
* [RFC9396 OAuth 2.0 Rich Authorization Requests](https://datatracker.ietf.org/doc/html/rfc9396)
* [RFC9449 OAuth 2.0 Demonstrating Proof of Possession (DPoP)](https://datatracker.ietf.org/doc/html/rfc9449)
* [RFC9700 Best Current Practice for OAuth 2.0 Security](https://datatracker.ietf.org/doc/html/rfc9700)
* [draft OAuth 2.0 for Browser-Based Applications](https://datatracker.ietf.org/doc/html/draft-ietf-oauth-browser-based-apps)
* [draft The OAuth 2.1 Authorization Framework](https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-12)

OpenID Connect について詳しくは以下の情報を参照してください。

* [OpenID Connect Core 1.0](https://openid.net/specs/openid-connect-core-1_0.html)
* [FAPI 2.0 Security Profile](https://openid.net/specs/fapi-security-profile-2_0-final.html)

# V11 暗号化

## 管理目標

本章の目標は、暗号の一般的な使用に関するベストプラクティスを定義するとともに、暗号の原則に関する基本的な理解を浸透させ、より耐性があり現代的なアプローチへの転換を促すことです。以下のことを推奨します。

* 安全に失敗し、進化する脅威に適応し、将来においても有効な堅牢な暗号システムを実装すること。
* 安全で業界のベストプラクティスに準拠した暗号メカニズムを利用すること。
* 適切なアクセス制御と監査を備えた安全な暗号鍵管理システムを維持すること。
* 暗号状況を定期的に評価して、新しいリスクを評価し、それに応じてアルゴリズムを適応すること。
* アプリケーションのライフサイクル全体を通して暗号ユースケースを検出して管理し、すべての暗号資産が把握されて保護されていることを確保すること。

一般的な原則とベストプラクティスの概説に加えて、このドキュメントでは付録 C - 暗号化標準の要件に関するより詳細な技術情報も提供します。これにはこの章の要件の目的として「承認済み」とみなされるアルゴリズムとモードを含みます。

シークレット管理や通信セキュリティなど、別の問題を解決するために暗号を使用する要件は、標準の別の部分にあります。

## V11.1 暗号インベントリとドキュメント

アプリケーションはデータ資産を分類に従って保護するために、強力な暗号アーキテクチャで設計する必要があります。すべてを暗号化することは無駄であり、なにも暗号化しないことは法的に過失となります。通常、アーキテクチャ設計や高レベル設計、デザインスプリント、アーキテクチャスパイクの際に、バランスをとる必要があります。暗号を「その場しのぎ」で設計したり、後から追加することは、単に最初から組み込むよりも、安全に実装するために必然的により多くのコストがかかります。

すべての暗号資産を定期的に発見し、インベントリ化し、評価することが重要です。その方法の詳細については付録を参照してください。

量子コンピューティングの台頭に備えて、将来を見据えた暗号システムの必要性も極めて重要です。ポスト量子暗号 (Post-Quantum Cryptography, PQC) は、RSA や 楕円曲線暗号 (ECC) などの広く使用されているアルゴリズムを破ることが予想される量子コンピュータによる攻撃に対して安全性を維持するように設計された暗号アルゴリズムを指します。

審査済みの PQC プリミティブと標準に関する最新のガイダンスについては付録を参照してください。

| # | 説明 | レベル |
| --- | --- | --- |
| **11.1.1** | NIST SP 800-57 などの鍵管理標準に準拠した暗号鍵の管理と暗号鍵のライフサイクルに関するポリシーが文書化されている。これには鍵が過剰共有 (たとえば、共有シークレットでは二つより多いエンティティ、秘密鍵では一つより多いエンティティ) されないようにすることも含む必要がある。 | 2 |
| **11.1.2** | 暗号インベントリは、実行され、維持され、定期的に更新され、アプリケーションで使用されるすべての暗号鍵、アルゴリズム、証明書を含む。また、システム内で鍵を使用できる場所とできない場所、および鍵を使用して保護できるデータと保護できないデータの種類を文書化する必要がある。 | 2 |
| **11.1.3** | 暗号検出メカニズムが採用され、暗号化、ハッシュ化、署名操作など、システム内のすべての暗号インスタンスを識別している。 | 3 |
| **11.1.4** | 暗号インベントリが維持されている。これには、将来の脅威に対応するために、ポスト量子暗号などの新しい標準への移行パスを概説する、文書化された計画を含む必要がある。 | 3 |

## V11.2 安全な暗号の実装

このセクションではアプリケーションの中核となる暗号アルゴリズムの選択、実装、および継続的な管理の要件を定義します。その目的は、現行の標準 (NIST, ISO/IEC など) やベストプラクティスに沿って、堅牢で業界で認められている暗号プリミティブのみが導入されるようにすることです。組織は各暗号コンポーネントがピアレビューされた証跡と実際のセキュリティテストに基づいて選択されるようにする必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **11.2.1** | 暗号操作には業界で検証済みの実装 (ライブラリやハードウェアアクセラレーション実装を含む) が使用される。 | 2 |
| **11.2.2** | アプリケーションは、乱数、認証された暗号化、MAC、ハッシュアルゴリズム、鍵長、ラウンド、暗号やモードをいつでも再構成、アップグレード、または交換できるように、暗号の敏捷性を考慮して設計され、暗号解読から保護している。同様に、鍵とパスワードを置換してデータを再暗号化することも可能である必要がある。これにより、承認済みポスト量子暗号 (PQC) スキームや標準の高保証実装が広く利用可能になると、PQC へのシームレスなアップグレードが可能になる。 | 2 |
| **11.2.3** | すべての暗号プリミティブは、アルゴリズム、鍵サイズ、構成に基づいて、最低でも 128 ビットのセキュリティを利用している。たとえば、256 ビットの ECC 鍵はおよそ 128 ビットのセキュリティを提供するが、RSA では 128 ビットのセキュリティを実現するために 3072 ビットの鍵を必要とする。 | 2 |
| **11.2.4** | 情報の漏洩を防ぐために、比較、計算、リターンの際に「短絡」操作がなく、すべての暗号操作は一定時間となっている。 | 3 |
| **11.2.5** | すべての暗号モジュールは失敗した場合の安全対策が施されており、パディングオラクル攻撃などの脆弱性を許さない方法でエラーが処理される。 | 3 |

## V11.3 暗号アルゴリズム

AES や CHACHA20 に基づいて構築された、認証された暗号アルゴリズムは現代の暗号プラクティスのバックボーンを形成します。

| # | 説明 | レベル |
| --- | --- | --- |
| **11.3.1** | 安全でないブロックモード (ECB など) や脆弱なパディングスキーム (PKCS#1 v1.5 など) を使用していない。 | 1 |
| **11.3.2** | AES with GCM などの承認済み暗号とモードのみを使用している。 | 1 |
| **11.3.3** | 暗号化されたデータは、できれば承認済みで認証済みの暗号方式を使用するか、承認済み暗号方式と承認済み MAC アルゴリズムを組み合わせることによって、認可されていない改変から保護している。 | 2 |
| **11.3.4** | ナンス、初期化ベクトル、その他の使い捨て番号は複数の暗号鍵とデータ要素のペアに使用していない。生成手法は、使用されるアルゴリズムに適したものである必要がある。 | 3 |
| **11.3.5** | 暗号アルゴリズムと MAC アルゴリズムの組み合わせは encrypt-then-MAC モードで動作している。 | 3 |

## V11.4 ハッシュ化とハッシュベース関数

暗号ハッシュは、デジタル署名、HMAC、鍵導出関数 (KDF)、ランダムビット生成、パスワードストレージなど、さまざまな暗号プロトコルで使用されます。暗号システムのセキュリティは、使用される基盤となるハッシュ関数の強度によって決まります。このセクションでは暗号操作で安全なハッシュ関数を使用するための要件を概説します。

パスワードの保存については、暗号化の付録と同様に、[OWASP Password Storage Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#password-hashing-algorithms) も役に立つコンテキストとガイダンスを提供します。

| # | 説明 | レベル |
| --- | --- | --- |
| **11.4.1** | デジタル署名、HMAC、KDF、ランダムビット生成など、一般的な暗号ユースケースには、承認済みハッシュ関数のみが使用されている。MD5 などの許可されていないハッシュ関数はいかなる暗号目的にも使用してはならない。 | 1 |
| **11.4.2** | パスワードは、現在のガイダンスに基づいて設定されたパラメータを用いた、承認済み計算集約型の鍵導出関数 (「パスワードハッシュ関数」とも呼ばれる) を使用して保存されている。この設定は、セキュリティとパフォーマンスのバランスをとり、必要なセキュリティレベルに対してブルートフォース攻撃を十分に困難にすべきである。 | 2 |
| **11.4.3** | データ認証やデータ完全性の一部としてデジタル署名に使用されるハッシュ関数は衝突耐性があり、適切なビット長を有している。衝突耐性が必要な場合、出力長は少なくとも 256 ビットである必要がある。第二原像攻撃攻撃への耐性のみが必要な場合、出力長は少なくとも 128 ビットである必要がある。 | 2 |
| **11.4.4** | アプリケーションは、パスワードから秘密鍵を導出する際に、鍵ストレッチパラメータを備えた承認済み鍵導出関数を使用している。使用するパラメータは、ブルートフォース攻撃によって生成された暗号鍵が侵害されることを防ぐために、セキュリティとパフォーマンスのバランスをとる必要がある。 | 2 |

## V11.5 乱数値

暗号論的に安全な疑似乱数生成 (Cryptographically Secure Pseudo-random Number Generation, CSPRNG) を正しく行うことは非常に困難です。一般的にシステム内のエントロピーの優れたソースは使い過ぎるとすぐに枯渇してしまいますが、ランダム性の少ないソースは鍵やシークレットが予測可能となる可能性があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **11.5.1** | 推測不可能であることを意図したすべての乱数と文字列は、暗号論的に安全な疑似乱数生成器 (CSPRNG) を使用して生成され、少なくとも 128 ビットのエントロピーを持つ必要がある。UUID はこの条件を満たさないことに注意する。 | 2 |
| **11.5.2** | 使用されている乱数生成メカニズムは、負荷が高い場合でも安全に機能するように設計されている。 | 3 |

## V11.6 公開鍵暗号

公開鍵暗号は複数の当事者間で秘密鍵を共有することが不可能または望ましくない場合に使用されます。

その一環として、暗号システムが現代の脅威に対して安全であり続けることを確保するには、Diffie-Hellman and Elliptic Curve Diffie-Hellman (ECDH) などの承認済み鍵交換メカニズムが必要です。「安全な通信」の章では TLS の要件について説明しているため、このセクションの要件は TLS 以外のユースケースで公開鍵暗号が使用されている状況を対象としています。

| # | 説明 | レベル |
| --- | --- | --- |
| **11.6.1** | 鍵の生成とシード、およびデジタル署名の生成と検証には、承認済みの暗号アルゴリズムと動作モードのみが使用される。鍵生成アルゴリズムはフェルマー因数分解に対して脆弱な RSA 鍵など、既知の攻撃に対して脆弱な安全でない鍵を生成してはいけない。 | 2 |
| **11.6.2** | 承認済み暗号アルゴリズム (Diffie-Hellman など) を鍵交換に使用し、鍵交換メカニズムが安全なパラメータを使用することに重点を置いている。これにより、Adversary-in-the-Middle 攻撃や暗号解読につながる可能性のある鍵確立プロセスへの攻撃を防ぐ。 | 3 |

## V11.7 使用中のデータの暗号化

処理中のデータを保護することが最も重要です。フルメモリ暗号化、転送中の暗号化、使用後のデータの可能な限り迅速な暗号化などの技法が推奨されます。

| # | 説明 | レベル |
| --- | --- | --- |
| **11.7.1** | フルメモリ暗号化を使用して、使用中の機密データを保護し、認可されていないユーザやプロセスによるアクセスを防いでいる。 | 3 |
| **11.7.2** | データの最小化により、処理中に露出するデータの量を最小限に抑え、使用後すぐに、または可能な限り速やかにデータが暗号化されることを確保している。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP Web Security Testing Guide: Testing for Weak Cryptography](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography)
* [OWASP Cryptographic Storage Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html)
* [FIPS 140-3](https://csrc.nist.gov/pubs/fips/140-3/final)
* [NIST SP 800-57](https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final)

# V12 安全な通信

## 管理目標

この章には、エンドユーザクライアントとバックエンドサービス間、および内部サービスとバックエンドサービス間の両方で、転送時のデータを保護するために導入すべき具体的なメカニズムに関する要件を含みます。

この章で推進される一般的な概念は以下のとおりです。

* 通信が外部で暗号化され、理想的には内部でも暗号化されていることを確保すること。
* 優先アルゴリズムや暗号方式など、最新のガイダンスを使用して暗号メカニズムを構成すること。
* 署名された証明書を使用することで、通信が不正な第三者に傍受されていないことを確保すること。

ASVS では、一般的な原則とベストプラクティスを概説するだけでなく、付録 C - 暗号化標準で暗号強度に関するより詳細な技術情報も提供しています。

## V12.1 一般的な TLS セキュリティガイダンス

このセクションは TLS 通信を保護する方法についての最初のガイダンスを提供します。最新のツールを使用して TLS 構成を継続的にレビューする必要があります。

ワイルドカード TLS 証明書の使用は本質的に安全でないわけではありませんが、所有するすべての環境 (実稼働、ステージング、開発、テストなど) にわたって展開される証明書の侵害は、それを使用するアプリケーションのセキュリティ態勢の侵害につながる可能性があります。適切な保護、管理を実施し、可能であれば、環境ごとに個別の TLS 証明書の使用を採用する必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **12.1.1** | TLS 1.2 や TLS 1.3 など、最新の推奨バージョンの TLS プロトコルのみが有効である。最新バージョンの TLS プロトコルを優先オプションにする必要がある。 | 1 |
| **12.1.2** | 推奨暗号スイートのみが有効であり、最も強力な暗号スイートが優先的に設定されている。L3 アプリケーションは前方秘匿性を提供する暗号スイートのみをサポートする必要がある。 | 2 |
| **12.1.3** | アプリケーションは、認証または認可に証明書 ID を使用する前に、mTLS クライアント証明書が信頼できることを検証している。 | 2 |
| **12.1.4** | Online Certificate Status Protocol (OCSP) Stapling などの証明書の失効を適切にチェックできる機能を設定し、有効にしている。 | 3 |
| **12.1.5** | アプリケーションの TLS 設定で Encrypted Client Hello (ECH) を有効にしており、TLS ハンドシェイクプロセス時に Server Name Indication (SNI) などの機密性の高いメタデータの開示を防いでいる。 | 3 |

## V12.2 外部向けサービスとの HTTPS 通信

アプリケーションが公開する外部向けサービスに対するすべての HTTP トラフィックが、公的に信頼できる証明書により暗号化されて送信されることを確認します。

| # | 説明 | レベル |
| --- | --- | --- |
| **12.2.1** | TLS がクライアントと外部向けの HTTP ベースのサービス間のすべての接続に使用されており、安全でない通信や非暗号化通信にフォールバックしない。 | 1 |
| **12.2.2** | 外部向けサービスが公的に信頼できる TLS 証明書を使用している。 | 1 |

## V12.3 一般的なサービス間通信セキュリティ

サーバ通信 (内部および外部の両方) は HTTP だけではありません。他のシステムとの接続も安全である必要があり、理想的には TLS を使用します。

| # | 説明 | レベル |
| --- | --- | --- |
| **12.3.1** | 監視システム、管理ツール、リモートアクセス、SSH、ミドルウェア、データベース、メインフレーム、パートナーシステム、外部 API など、アプリケーションとのすべてのインバウンドおよびアウトバウンドの接続に、TLS などの暗号化プロトコルが使用されている。サーバは安全でないプロトコルや暗号化されていないプロトコルにフォールバックしてはいけない。 | 2 |
| **12.3.2** | TLS クライアントは TLS サーバと通信する前に受信した証明書を検証している。 | 2 |
| **12.3.3** | アプリケーション内の HTTP ベースの内部サービス間のすべての接続に TLS または別の適切なトランスポート暗号化メカニズムが使用されており、安全でない通信や暗号化されていない通信にフォールバックしていない。 | 2 |
| **12.3.4** | 内部サービス間の TLS 接続には信頼できる証明書が使用されている。内部で生成された証明書または自己署名証明書を使用する場合、受け側のサービスは特定の内部認証局や特定の自己署名証明書のみを信頼するように構成する必要がある。 | 2 |
| **12.3.5** | システム内で通信するサービス (サービス内通信) は強力な認証を使用して、各エンドポイントが検証されていることを確保している。TLS クライアント認証などの強力な認証方式を採用し、公開鍵インフラストラクチャとリプレイ攻撃に耐性のあるメカニズムを使用して、アイデンティティを確認する必要がある。マイクロサービスアーキテクチャの場合は、サービスメッシュを使用して証明書管理を簡素化し、セキュリティを強化することを検討する。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP - Transport Layer Security Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html)
* [Mozilla's Server Side TLS configuration guide](https://wiki.mozilla.org/Security/Server_Side_TLS)
* [Mozilla's tool to generate known good TLS configurations](https://ssl-config.mozilla.org/)
* [O-Saft - OWASP Project to validate TLS configuration](https://owasp.org/www-project-o-saft/)

# V13 構成

## 管理目標

アプリケーションのデフォルト構成はインターネット上での使用に安全である必要があります。

この章では開発、ビルド、デプロイメント時に適用されるものを含む、これを実現するために必要なさまざまな構成に関するガイダンスを提供します。

カバーされるトピックには、データ漏洩の防止、コンポーネント間の通信の安全な管理、シークレットの保護などを含みます。

## V13.1 構成ドキュメント

このセクションでは、アプリケーションが内部および外部のサービスと通信する方法についてと、これらのサービスにアクセスできないことによる可用性の損失を防ぐための技法についてのドキュメント要件を提供します。また、シークレットに関するドキュメントも扱います。

| # | 説明 | レベル |
| --- | --- | --- |
| **13.1.1** | アプリケーションのすべての通信が文書化されている。これには、アプリケーションが依存する外部サービスや、アプリケーションが接続する外部ロケーションをエンドユーザが提供できる可能性がある場合を含む必要がある。 | 2 |
| **13.1.2** | アプリケーションが使用する各サービスについて、ドキュメントでは同時接続最大数 (接続プールの制限など) と、その制限に達した際のアプリケーションの動作 (フォールバックやリカバリのメカニズムを含む) を定義し、サービス拒否状態を防いでいる。 | 3 |
| **13.1.3** | アプリケーションドキュメントでは、使用するすべての外部システムやサービス (データベース、ファイルハンドル、スレッド、HTTP 接続) のリソース管理戦略を定義している。これには、リソース解放手順、タイムアウト設定、障害処理、再試行ロジックが実装されている場合、再試行制限、遅延、バックオフアルゴリズムの指定を含む。同期 HTTP リクエストレスポンス操作では、短いタイムアウトを義務付け、再試行を無効にするか、再試行回数を厳しく制限して、連鎖的な遅延やリソース枯渇を防ぐ必要がある。 | 3 |
| **13.1.4** | アプリケーションのドキュメントでは、組織の脅威モデルとビジネス要件に基づいて、アプリケーションのセキュリティにとって重要なシークレットとそれらを入れ替えるスケジュールを定義している。 | 3 |

## V13.2 バックエンド通信構成

アプリケーションは、API、データベース、その他のコンポーネントを含む複数のサービスとやり取りします。これらはアプリケーションの内部にあるとみなされるかもしれませんが、アプリケーションの標準アクセス制御メカニズムに含まれていないこともあれば、完全に外部にあることもあります。いずれの場合も、これらのコンポーネントと安全にやり取りするようにアプリケーションを構成し、必要に応じて、その構成を保護する必要があります。

注: 「安全な通信」の章では転送時の暗号化に関するガイダンスを提供しています。

| # | 説明 | レベル |
| --- | --- | --- |
| **13.2.1** | API、ミドルウェア、データレイヤなど、アプリケーションの標準ユーザセッションメカニズムをサポートしていないバックエンドアプリケーションコンポーネント間の通信は認証されている。認証は、パスワード、API キー、特権アクセスを備えた共有アカウントなどの不変のクレデンシャルではなく、個別のサービスアカウント、短期トークン、証明書ベースの認証を使用する必要がある。 | 2 |
| **13.2.2** | ローカルまたはオペレーティングシステムのサービス、API、ミドルウェア、データレイヤなどのバックエンドアプリケーションコンポーネント間の通信は最小限の権限が割り当てられたアカウントで実行されている。 | 2 |
| **13.2.3** | サービス認証にクレデンシャルを使用する必要がある場合、コンシューマが使用するデフォルトクレデンシャル (root/root や admin/admin など) ではない。 | 2 |
| **13.2.4** | 許可リストを使用して、アプリケーションが通信 (アウトバウンドリクエスト、データロード、ファイルアクセスなど) を許可される外部のリソースやシステムを定義している。この許可リストは、アプリケーション層、Web サーバ、ファイアウォール、または複数の層の組み合わせで実装できる。 | 2 |
| **13.2.5** | Web サーバまたはアプリケーションサーバはサーバがリクエストを送信したり、データやファイルをロードできるリソースやシステムの許可リストで構成されている。 | 2 |
| **13.2.6** | アプリケーションが個別のサービスに接続する場合、最大並列接続数、最大許容接続数に達した際の動作、接続タイムアウト、再試行戦略など、各接続について文書化された構成に従っている。 | 3 |

## V13.3 シークレット管理

シークレット管理は、アプリケーションで使用されるデータの保護を確保するために不可欠な構成タスクです。暗号に関する具体的な要件は「暗号化」の章にありますが、このセクションではシークレットの管理と取り扱いの側面に焦点を当てています。

| # | 説明 | レベル |
| --- | --- | --- |
| **13.3.1** | key vault などのシークレット管理ソリューションは、バックエンドシークレットを安全に作成、保管、アクセス制御、破棄するために使用している。これには、パスワード、鍵マテリアル、データベースやサードパーティシステムとの統合、時間ベースのトークンのための鍵やシード、その他の内部セキュリティ、API キーなどを含む。シークレットはアプリケーションのソースコードやビルド成果物に含めてはいけない。L3 アプリケーションでは、これには HSM などのハードウェア支援のソリューションを含む必要がある。 | 2 |
| **13.3.2** | シークレット資産へのアクセスは最小権限の原則に従っている。 | 2 |
| **13.3.3** | すべての暗号操作は、隔離されたセキュリティモジュール (vault やハードウェアセキュリティモジュールなど) を使用して実行され、鍵マテリアルがセキュリティモジュールの外部へ漏れないように安全に管理および保護している。 | 3 |
| **13.3.4** | シークレットはアプリケーションのドキュメントに基づいて有効期限が切れて入れ替えるように構成されている。 | 3 |

## V13.4 意図しない情報漏洩

不要なデータの開示を避けるために、本番用の構成を堅牢化する必要があります。これらの問題の多くは重大なリスクとして評価されることはほとんどありませんが、他の脆弱性と連鎖することがよくあります。これらの問題がデフォルトで存在しない場合、アプリケーションを攻撃するハードルが上がります。

たとえば、サーバサイドコンポーネントのバージョンを非表示にしても、すべてのコンポーネントにパッチを適用する必要性がなくなるわけではありませんし、フォルダの一覧表示を無効にしても、認可コントロールを使用したり、パブリックフォルダからファイルを遠ざける必要性がなくなるわけではありませんが、ハードルが上がります。

| # | 説明 | レベル |
| --- | --- | --- |
| **13.4.1** | アプリケーションは .git や .svn フォルダなどのソース管理メタデータなしでデプロイされるか、またはこれらのフォルダが外部からもアプリケーション自体からもアクセスできない方法でデプロイされる。 | 1 |
| **13.4.2** | デバッグ機能の露出や情報漏洩を防ぐため、本番環境ではすべてのコンポーネントでデバッグモードが無効になっている。 | 2 |
| **13.4.3** | Web サーバは、明示的に意図されていない限り、ディレクトリリストをクライアントに公開していない。 | 2 |
| **13.4.4** | 潜在的な情報漏洩を避けるため、本番環境では HTTP TRACE メソッドの使用はサポートされない。 | 2 |
| **13.4.5** | ドキュメント (内部 API など) および監視エンドポイントは明示的に意図されない限り公開していない。 | 2 |
| **13.4.6** | アプリケーションはバックエンドコンポーネントの詳細なバージョン情報を公開していない。 | 3 |
| **13.4.7** | 意図しない情報、構成、ソースコードの漏洩を防ぐために、Web 層が特定のファイル拡張子を持つファイルのみを処理するように構成されている。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP Web Security Testing Guide: Configuration and Deployment Management Testing](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing)

# V14 データ保護

## 管理目標

アプリケーションはすべての使用パターンとユーザの行動を考慮することはできないため、クライアントデバイス上の機密データへの不正アクセスを制限するためのコントロールを実装する必要があります。

この章には、どのようなデータを保護する必要があるのか、どのように保護すべきなのか、実装すべき具体的なメカニズム、回避すべき落とし穴といった定義に関連する要件を含んでいます。

データ保護に関するもう一つの懸念事項は、一括抽出、変更、過度の使用です。各システムの要件は大きく異なる可能性が高いため、「異常」の判断には脅威モデルとビジネスリスクを考慮する必要があります。ASVS の観点からは、これらの問題の検出は「セキュリティログ記録とエラー処理」の章で扱われ、制限の設定は「バリデーションとビジネスロジック」の章で扱われます。

## V14.1 データ保護ドキュメント

データ保護を実現するための重要な前提条件は、どのデータを機密とみなすべきかを分類することです。機密性にはいくつかのレベルがあり、そのレベルごとに、そのレベルのデータを保護するために必要なコントロールが異なります。

さまざまなプライバシー規制や法律があり、機密性の高い個人情報の保存、使用、および転送にアプリケーションがどのようにアプローチする必要があるかに影響します。このセクションではこのような種類のデータ保護やプライバシー法と重複して説明するようなことはせず、機密データを保護するための重要な技術的留意点に焦点を当てます。地域の法律や規制を調べ、必要に応じて資格のあるプライバシーの専門家または弁護士に相談してください。

| # | 説明 | レベル |
| --- | --- | --- |
| **14.1.1** | アプリケーションにより作成および処理されるすべての機密データが特定され、保護レベルに分類されている。これは Base64 や JWT 内のプレーンテキストペイロードなど、エンコードされているだけで簡単にデコードできるデータを含む。保護レベルでは、アプリケーションが準拠する必要のあるデータ保護やプライバシー規制を考慮する必要がある。 | 2 |
| **14.1.2** | 文書化された保護要件一式がすべての機密データ保護レベルにある。これには一般的な暗号化、完全性検証、リテンション、データのログ記録方法、ログ内の機密データに関するアクセス制御、データベースレベルの暗号化、プライバシー、使用されるプライバシー強化技法に関連する要件、およびその他の機密性要件を含む必要がある (ただし、これらに限定されない)。 | 2 |

## V14.2 一般的なデータ保護

このセクションにはデータの保護に関連するさまざまな実用的な要件を含みます。その多くは意図しないデータ漏洩などの特定の問題に固有のものですが、各データ項目に必要な保護レベルに基づいて保護コントロールを実装するという一般的な要件もあります。

| # | 説明 | レベル |
| --- | --- | --- |
| **14.2.1** | 機密データが HTTP メッセージボディまたはヘッダフィールドでのみサーバに送信され、URL とクエリ文字列には API キーやセッショントークンなどの機密データが含まれない。 | 1 |
| **14.2.2** | アプリケーションは機密データをロードバランサやアプリケーションキャッシュなどのサーバコンポーネントにキャッシュされないように防止している、もしくはデータを使用後に安全に削除している。 | 2 |
| **14.2.3** | アプリケーションの制御外で望ましくないデータ収集を防ぐために、定義された機密データは信頼できないパーティ (ユーザトラッカーなど) には送信されない。 | 2 |
| **14.2.4** | 暗号化、完全性検証、保持、データのログ記録方法、ログ内の機密データへのアクセス制御、プライバシー、プライバシー強化テクノロジに関連する機密データに関するコントロールは、特定のデータの保護レベルに対するドキュメントで定義されているとおりに実装されている。 | 2 |
| **14.2.5** | そのリソースの期待されるコンテンツタイプを持ち、機密性の高い動的コンテンツを含まないレスポンスのみをキャッシュするように、キャッシュメカニズムが設定されている。存在しないファイルがアクセスされる場合、Web サーバーは有効な別のファイルを返すのではなく 404 または 302 レスポンスを返す必要がある。これにより Web Cache Deception 攻撃を防ぐことができる。 | 3 |
| **14.2.6** | アプリケーションは、アプリケーションの機能に必要な最小限の機密データのみを返している。たとえば、クレジットカード番号の一部の数字のみを返しており、番号全体は返していない。完全なデータが必要な場合は、ユーザが明示的に表示しない限り、ユーザインタフェースでマスクする必要がある。 | 3 |
| **14.2.7** | 機密情報はデータ保持分類の対象となり、古くなったデータや不要なデータは、定められたスケジュールに従って、または状況に応じて自動的に削除される。 | 3 |
| **14.2.8** | ユーザーが保存に同意しない限り、ユーザーが送信したファイルのメタデータから機密情報が削除されている。 | 3 |

## V14.3 クライアントサイドのデータ保護

このセクションには、アプリケーションのクライアントサイドまたはユーザエージェントサイドで、特定の方法でデータが漏洩することを防ぐための要件を含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **14.3.1** | クライアントやセッションが終了した後、ブラウザ DOM などのクライアントストレージから認証データがクリアされる。'Clear-Site-Data' HTTP レスポンスヘッダフィールドで対応できるかもしれないが、セッションが終了した際にサーバ接続が利用できない場合にはクライアントサイドでもクリアできるようにする必要がある。 | 1 |
| **14.3.2** | 機密データがブラウザでキャッシュされないように、アプリケーションは十分なキャッシュ防止 HTTP レスポンスヘッダフィールド (つまり Cache-Control: no-store) を設定している。 | 2 |
| **14.3.3** | ブラウザストレージ (localStorage、sessionStorage、IndexedDB、クッキーなど) に保存されるデータに、セッショントークンを除いて、機密データが含まれていない。 | 2 |

## 参考情報

詳しくは以下の情報を参照してください。

* [Consider using the Security Headers website to check security and anti-caching header fields](https://securityheaders.com/)
* [Documentation about anti-caching headers by Mozilla](https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching)
* [OWASP Secure Headers project](https://owasp.org/www-project-secure-headers/)
* [OWASP Privacy Risks Project](https://owasp.org/www-project-top-10-privacy-risks/)
* [OWASP User Privacy Protection Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html)
* [Australian Privacy Principle 11 - Security of personal information](https://www.oaic.gov.au/privacy/australian-privacy-principles/australian-privacy-principles-guidelines/chapter-11-app-11-security-of-personal-information)
* [European Union General Data Protection Regulation (GDPR) overview](https://www.edps.europa.eu/data-protection_en)
* [European Union Data Protection Supervisor - Internet Privacy Engineering Network](https://www.edps.europa.eu/data-protection/ipen-internet-privacy-engineering-network_en)
* [Information on the "Clear-Site-Data" header](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Clear-Site-Data)
* [White paper on Web Cache Deception](https://www.blackhat.com/docs/us-17/wednesday/us-17-Gil-Web-Cache-Deception-Attack-wp.pdf)

# V15 セキュアコーディングとアーキテクチャ

## 管理目標

多くの ASVS 要件は、認証や認可といった特定のセキュリティ領域に関連しているか、ログ記録やファイル処理などの特定のタイプのアプリケーション機能に関係しています。

この章ではアプリケーションの設計と開発の際に考慮すべき、一般的なセキュリティ要件を提供します。これらの要件は、クリーンなアーキテクチャとコード品質だけでなく、アプリケーションセキュリティに必要となる特定のアーキテクチャとコーディングプラクティスについても焦点を当てています。

## V15.1 セキュアコーディングとアーキテクチャドキュメント

安全で防御力の高いアーキテクチャを確立するための要件の多くは、特定のセキュリティコントロールの実装とアプリケーション内で使用されるコンポーネントに関する決定を明確に文書化することに依存します。

このセクションでは、「危険な機能」を含むとみなされるコンポーネントや「リスクのあるコンポーネント」と考えられるコンポーネントの特定など、ドキュメント要件について概説します。

「危険な機能」を持つコンポーネントとは、信頼できないデータのデシリアライゼーション、生ファイルやバイナリデータの解析、動的コード実行、直接メモリ操作などの操作を実行する、内部開発またはサードパーティのコンポーネントを指します。これらのタイプの操作の脆弱性は、アプリケーションを侵害し、基盤となるインフラストラクチャを開示する可能性のある高いリスクをもたらします。

「リスクのあるコンポーネント」とは、開発プロセスや機能に関するセキュリティコントロールが欠如しているか適切に実装されていないサードパーティライブラリ (つまり、内部開発ではないもの) を指します。たとえば、保守が不十分であったり、サポートされていなかったり、サポート終了段階にあったり、重大な脆弱性の経歴があるコンポーネントなどがあります。

また、このセクションでは、サードパーティコンポーネントの脆弱性に対処するための適切な期間を定義することの重要性も強調しています。

| # | 説明 | レベル |
| --- | --- | --- |
| **15.1.1** | アプリケーションドキュメントでは、脆弱性があるサードパーティコンポーネントのバージョンやライブラリ全般の更新について、リスクに基づく修復時間枠を定義して、これらのコンポーネントからのリスクを最小限に抑えている。 | 1 |
| **15.1.2** | 使用しているすべてのサードパーティライブラリについて、ソフトウェア部品表 (SBOM) などのインベントリカテゴリを保守している。コンポーネントは事前定義済みで信頼でき、継続的に保守されているリポジトリから取得している。 | 2 |
| **15.1.3** | アプリケーションドキュメントでは、時間のかかる機能やリソースを必要とする機能を特定している。これには、この機能の過剰使用による可用性の損失を防ぐ方法や、レスポンスの構築にコンシューマのタイムアウトよりも長い時間がかかる状況を避ける方法を含む必要がある。可能性のある防御策としては、非同期処理、キューの使用、ユーザごとおよびアプリケーションごとの並列処理の制限などがある。 | 2 |
| **15.1.4** | アプリケーションドキュメントでは「リスクのあるコンポーネント」とみなされるサードパーティライブラリを強調している。 | 3 |
| **15.1.5** | アプリケーションドキュメントでは「危険な機能」が使用されているアプリケーションのパーツを強調している。 | 3 |

## V15.2 セキュリティアーキテクチャと依存関係

このセクションでは、依存関係管理を通じてリスクのある、古い、または安全でない依存関係とコンポーネントを処理するための要件を含みます。

また、サンドボックス化、カプセル化、コンテナ化、ネットワーク分離などのアーキテクチャレベルの技法を使用して、「危険な操作」や「リスクのあるコンポーネント」 (前のセクションで定義) の使用による影響を軽減し、リソースを必要とする機能の過剰使用による可用性の損失を防ぐことも含みます。

| # | 説明 | レベル |
| --- | --- | --- |
| **15.2.1** | アプリケーションは、文書化された更新および修復の時間枠に違反していないコンポーネントのみを含む。 | 1 |
| **15.2.2** | アプリケーションは、これに関する文書化されたセキュリティ上の決定と戦略に基づいて、時間のかかる機能やリソースを必要とする機能による可用性の損失に対する防御策を実装している。 | 2 |
| **15.2.3** | 本番環境にはアプリケーションの機能に必要な機能のみを含み、テストコード、サンプルスニペット、開発機能などの不要な機能を公開していない。 | 2 |
| **15.2.4** | サードパーティコンポーネントとそのすべての推移的依存関係は、内部所有であるか外部ソースであるかに関わらず、想定されたリポジトリから組み込まれており、依存関係攪乱攻撃のリスクはない。 | 3 |
| **15.2.5** | アプリケーションは、「危険な機能」を含む、あるいは「リスクのあるコンポーネント」とみなされるサードパーティライブラリを使用していると文書化されているアプリケーションのパーツに対して、追加の保護を実装している。これは、サンドボックス化、カプセル化、コンテナ化、ネットワークレベルの分離などの技法を含み、アプリケーションの一部分を侵害した攻撃者がアプリケーションの他の部分に侵入するのを遅らせたり阻止する。 | 3 |

## V15.3 防御的コーディング

このセクションでは、型ジャグリング、プロトタイプ汚染など、特定の言語における安全でないコーディングパターンの使用に起因する脆弱性の種類について説明します。一部はすべての言語には関係しないかもしれない一方で、他のものは言語固有の修正があったり、特定の言語やフレームワークが HTTP パラメータなどの機能を処理する方法に関係するかもしれません。また、アプリケーションの更新を暗号的に検証しないことのリスクも考慮します。

また、オブジェクトを使用してデータ項目を表現し、外部 API を介してこれらを受け入れ、返す際に関連するリスクも考慮します。この場合、アプリケーションは書き込み不可のデータフィールドがユーザ入力 (マスアサインメント) によって変更されないことと、API が返されるデータフィールドについて選択的であることを確保する必要があります。フィールドへのアクセスがユーザのパーミッションに依存する場合、これは認可の章にあるフィールドレベルのアクセス制御要件のコンテキストで考慮する必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **15.3.1** | アプリケーションはデータオブジェクトからフィールドの必要なサブセットのみを返す。たとえば、個別のフィールドの中にはユーザーがアクセスすべきではないものがあるため、データオブジェクト全体を返すべきではない。 | 1 |
| **15.3.2** | アプリケーションバックエンドが外部 URL を呼び出す場合、意図した機能でない限りリダイレクトに従わないように設定されている。 | 2 |
| **15.3.3** | アプリケーションには、コントローラやアクションごとに許可されるフィールドを制限することで、マスアサインメント攻撃から保護する対策がある。たとえば、そのアクションの一部となることを意図していないフィールド値を挿入または更新することはできない。 | 2 |
| **15.3.4** | すべてのプロキシとミドルウェアコンポーネントは、エンドユーザが操作できない信頼できるデータフィールドを使用してユーザのオリジナル IP アドレスを正しく転送し、アプリケーションと Web サーバは、動的 IP、VPN、企業のファームウェアによってオリジナル IP アドレスが信頼できない可能性があることを考慮して、ログ記録や、レート制限などのセキュリティ決定にこの正しい値を使用している。 | 2 |
| **15.3.5** | アプリケーションは変数が正しい型であることを明示的に保証し、厳密な等価演算と比較演算を実行している。これは、アプリケーションコードが変数の型について仮定することによって引き起こされる型ジャグリングや型コンフュージョンの脆弱性を回避するためである。 | 2 |
| **15.3.6** | JavaScript コードはオブジェクトリテラルの代わりに Set() や Map() を使用するなど、プロトタイプ汚染を防ぐ方法で記述している。 | 2 |
| **15.3.7** | アプリケーションが HTTP 変数汚染攻撃に対する防御策を備えている。特にアプリケーションフレームワークがリクエストパラメータのソース (クエリ文字列、ボディパラメータ、クッキー、ヘッダフィールド) を区別しない場合はこの防御が必要である。 | 2 |

## V15.4 安全な同時並行性

競合状態、チェック時間と使用時間 (TOCTOU) の脆弱性、デッドロック、ライブロック、スレッド枯渇、不適切な同期などの同時並行性の問題は、予期しない動作やセキュリティリスクにつながる可能性があります。このセクションではこれらのリスクを緩和するためのさまざまな技法と戦略を説明します。

| # | 説明 | レベル |
| --- | --- | --- |
| **15.4.1** | マルチスレッドコード内の共有オブジェクト (キャッシュ、ファイル、複数のスレッドによってアクセスするメモリ内オブジェクトなど) は、スレッドセーフな型と、ロックやセマフォなどの同期メカニズムを使用して安全にアクセスして、競合状態やデータ破損を回避している。 | 3 |
| **15.4.2** | リソースの存在やパーミッションなどのリソースの状態のチェックとそれらに依存するアクションは、単一のアトミック操作として実行され、チェック時間と使用時間 (TOCTOU) の競合状態を防いでいる。たとえば、ファイルを開く前にそのファイルが存在するかどうかをチェックしたり、ユーザのアクセスを付与する前に検証している。 | 3 |
| **15.4.3** | ロックは一貫して使用され、互いに待機したり無限に再試行することによってスレッドがスタックすることを避けている。また、ロックロジックはリソースの管理を担当するコード内にとどめ、ロックが外部のクラスやコードによって不注意に、あるいは悪意を持って変更されないようにしている。 | 3 |
| **15.4.4** | リソース割り当てのポリシーは、スレッドプールを活用するなどしてリソースへの公平なアクセスを確保し、優先度の低いスレッドが妥当な時間枠内で処理を進められるようにすることで、スレッドの枯渇を防いでいる。 | 3 |

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP Prototype Pollution Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Prototype_Pollution_Prevention_Cheat_Sheet.html)
* [OWASP Mass Assignment Prevention Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html)
* [OWASP CycloneDX Bill of Materials Specification](https://owasp.org/www-project-cyclonedx/)
* [OWASP Web Security Testing Guide: Testing for HTTP Parameter Pollution](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/04-Testing_for_HTTP_Parameter_Pollution)

# V16 セキュリティログ記録とエラー処理

## 管理目標

セキュリティログは、エラーログやパフォーマンスログとは異なり、認証の判定、アクセス制御の判定、入力バリデーションやビジネスロジックバリデーションなどのセキュリティコントロールのバイパスの試行など、セキュリティ関連イベントを記録するために使用されます。その目的は、SIEM などの分析ツールに高信号の構造化されたデータを提供することで、検知、対応、調査を支援することです。

法的に要求されない限り、ログは機密性の高い個人データを含むべきではなく、ログ記録されたデータは高価値資産として保護する必要があります。ログ記録はプライバシーやシステムのセキュリティを侵害してはいけません。また、アプリケーションは安全に失敗し、不必要な開示や中断を避ける必要もあります。

詳細な実装ガイダンスについては、参考情報セクションの OWASP Cheat Sheet を参照してください。

## V16.1 セキュリティログ記録ドキュメント

このセクションではアプリケーションスタック全体にわたるログ記録の明確で完全なインベントリを確保します。これは、効果的なセキュリティ監視、インシデント対応、コンプライアンスに不可欠です。

| # | 説明 | レベル |
| --- | --- | --- |
| **16.1.1** | アプリケーションのテクノロジスタックの各レイヤで実行されるログ記録、ログ記録されるイベント、ログ形式、ログ記録の保存場所、ログ記録の使用方法、ログ記録へのアクセスコントロール方法、ログの保存期間を文書化したインベントリが存在している。 | 2 |

## V16.2 一般的なログ記録

このセクションでは、セキュリティログが一貫して構造化され、期待されるメタデータを含むことを確保するための要件を提供します。その目標は、分散したシステムやツール間でログを機械可読かつ分析可能にすることです。

当然のことながら、セキュリティイベントには機密データを含むことがよくあります。そのようなデータが考慮なしにログ記録されると、ログ自体が機密扱いとなり、暗号要件、より厳密な保有ポリシー、監査時の開示の可能性の対象となります。

したがって、必要なものだけをログ記録して、ログデータを他の機密資産と同じように慎重に扱うことが重要です。

以下の要件は、ログ記録のメタデータ、同期、フォーマット、コントロールに関する基本的な要件を確立します。

| # | 説明 | レベル |
| --- | --- | --- |
| **16.2.1** | 各ログエントリにはイベント発生時のタイムラインを詳細に調査するために必要なメタデータ (いつ、どこで、誰が、何を、など) を含んでいる。 | 2 |
| **16.2.2** | すべてのログ記録コンポーネントのタイムソースが同期され、セキュリティイベントメタデータのタイムスタンプは UTC を使用するか、明示的なタイムゾーンオフセットを含む。分散システム間の一貫性を確保し、夏時間への移行時の混乱を防ぐため、UTC が推奨される。 | 2 |
| **16.2.3** | アプリケーションはログインベントリに文書化されているファイルおよびサービスのログだけを保存またはブロードキャストしている。 | 2 |
| **16.2.4** | ログは使用しているログプロセッサによって、できれば共通のログ形式を使用して、読み取りおよび関連付けできる。 | 2 |
| **16.2.5** | 機密データをログ記録する際、アプリケーションはデータの保護レベルに基づいてログ記録を実施している。たとえば、クレデンシャルや支払詳細など、特定のデータをログ記録することは許可できないことがある。セッショントークンなどのその他のデータは、ハッシュ化または、全体または一部がマスクすることによってのみログ記録できる。 | 2 |

## V16.3 セキュリティイベント

このセクションでは、アプリケーション内のセキュリティ関連イベントをログ記録するための要件を定義します。これらのイベントを捕捉することは、不審な動作の検出、調査のサポート、コンプライアンス義務の履行に不可欠です。

このセクションはログ記録する必要があるイベントの種類を概説しますが、網羅的な詳細を提供するものではありません。各アプリケーションには固有のリスク要因と運用上のコンテキストがあります。

ASVS にはセキュリティイベントのログ記録をスコープに含みますが、アラートと相関 (SIEM ルールや監視インフラストラクチャなど) はスコープ外とみなされ、運用システムと監視システムによって処理されることに注意してください。

| # | 説明 | レベル |
| --- | --- | --- |
| **16.3.1** | すべての認証操作は成功した試行と失敗した試行を含めてログ記録されている。使用された認証の種類や要素などの追加のメタデータも収集する必要がある。 | 2 |
| **16.3.2** | 失敗した認可の試行をログ記録している。L3 では、これは、機密データへのアクセス時のログ記録 (機密データ自体はログ記録しない) を含む、すべての認可判定のログ記録を含む必要がある。 | 2 |
| **16.3.3** | アプリケーションはドキュメントで定義されているセキュリティイベントをログ記録し、入力バリデーション、ビジネスロジック、アンチオートメーションなどのセキュリティ制御をバイパスする試みもログ記録している。 | 2 |
| **16.3.4** | アプリケーションは、予期しないエラーや、バックエンド TLS 障害などのセキュリティ制御障害をログ記録している。 | 2 |

## V16.4 ログ保護

ログはフォレンジックにおける貴重な資料であり、保護する必要があります。ログが簡単に変更または削除できる場合、ログの完全性が失われ、インシデント調査や法的手続きにおいて信頼性を失うことになります。ログは、内部アプリケーションの動作や機密性の高いメタデータを露出する可能性があり、攻撃者にとって魅力的な標的となります。

このセクションでは、ログが不正アクセス、改竄、開示から保護され、安全に分離されたシステムに安全に送信および保存することを確保するための要件を定義します。

| # | 説明 | レベル |
| --- | --- | --- |
| **16.4.1** | ログインジェクションを防ぐためにすべてのログ記録コンポーネントがデータを適切にエンコードしている。 | 2 |
| **16.4.2** | ログは不正なアクセスから保護されており、改変できない。 | 2 |
| **16.4.3** | 分析、検出、警告、エスカレーションのために、ログが論理的に分離されたシステムに安全に送信されている。これはアプリケーションが侵害されたとしても、ログが侵害されないことを保証することを目的としている。 | 2 |

## V16.5 エラー処理

このセクションでは、機密性の高い内部詳細を開示することなく、アプリケーションが正常かつ安全に失敗することを確保するための要件を定義します。

| # | 説明 | レベル |
| --- | --- | --- |
| **16.5.1** | 予期しないエラーやセキュリティ上重要なエラーが発生した際、スタックトレース、クエリ、秘密鍵、トークンなどの機密性の高い内部システムデータを開示しない、一般的なメッセージをコンシューマに返している。 | 2 |
| **16.5.2** | 外部リソースへのアクセスが失敗した場合でも、たとえば、サーキットブレーカーやグレースフルデグラデーションなどのパターンを使用することで、アプリケーションは安全に動作し続ける。 | 2 |
| **16.5.3** | 例外が発生した場合など、アプリケーションは適切かつ安全に失敗し、検証ロジックの結果としてのエラーにも関わらずトランザクションを処理するようなフェイルオープン状態を防いでいる。 | 2 |
| **16.5.4** | 未処理の例外をすべてキャッチする「最終手段」のエラーハンドラが定義されている。これは、ログファイルに記録する必要があるエラー詳細を失わないようにするためであり、エラーがアプリケーションプロセス全体を停止して可用性が失われないようにするためである。 | 3 |

注: 特定の言語 (Swift、Go、一般的なデザインプラクティスによって多くの関数型言語を含む) は、例外や最終手段イベントハンドラをサポートしていません。このような場合、アーキテクトと開発者は、パターン、言語、フレームワークに適した方法を使用して、例外イベント、予期しないイベント、セキュリティ関連のイベントをアプリケーションが安全に処理できるようにする必要があります。

## 参考情報

詳しくは以下の情報を参照してください。

* [OWASP Web Security Testing Guide: Testing for Error Handling](https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/08-Testing_for_Error_Handling/README)
* [OWASP Authentication Cheat Sheet section about error messages](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html#authentication-and-error-messages)
* [OWASP Logging Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html)
* [OWASP Application Logging Vocabulary Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Logging_Vocabulary_Cheat_Sheet.html)

# V17 WebRTC

## 管理目標

Web リアルタイム通信 (Web Real-Time Communication, WebRTC) は現代のアプリケーションにおいてリアルタイムの音声、動画、データ交換を可能にします。普及が進むにつれて、WebRTC インフラストラクチャのセキュリティ確保が重要になります。このセクションでは、WebRTC システムを開発、ホスト、統合する関係者向けのセキュリティ要件を提供します。

WebRTC 市場は三つのセグメントに大別できます。

1. 製品開発者: WebRTC 製品やソリューションを作成および提供するプロプライエタリおよびオープンソースのベンダーです。他者が使用できる堅牢で安全な WebRTC テクノロジの開発に重点を置いています。
2. サービスとしての通信プラットフォーム (Communication Platforms as a Service, CPaaS): WebRTC の機能を実現するために必要な API、SDK、インフラストラクチャやプラットフォームを提供します。CPaaS プロバイダは第一カテゴリの製品を使用することも、独自の WebRTC ソフトウェアを開発して、これらのサービスを提供することもできます。
3. サービスプロバイダ: 製品開発者や CPaaS プロバイダの製品を活用したり、独自の WebRTC ソリューションを開発する組織です。オンライン会議、ヘルスケア、e ラーニングなど、リアルタイム通信が重要な領域に対してアプリケーションを作成および実装します。

ここで説明するセキュリティ要件は主に以下のような製品開発者、CPaaS、サービスプロバイダを対象としています。

* オープンソースソリューションを利用して WebRTC アプリケーションを構築する。
* インフラストラクチャの一部として商用 WebRTC 製品を使用する。
* 内部で開発した WebRTC ソリューションを使用するか、さまざまなコンポーネントを統合して、まとまりのあるサービスを提供する。

これらのセキュリティ要件は CPaaS ベンダーが提供する SDK や API だけを使用する開発者には適用しないことに注意することが重要です。そのような開発者としては、一般的に CPaaS プロバイダがプラットフォーム内の基本的なセキュリティの懸念のほとんどに責任があり、ASVS のような汎用的なセキュリティ標準では開発者のニーズに完全に対応できないかもしれません。

## V17.1 TURN サーバ

このセクションでは、独自の TURN (Traversal Using Relays around NAT) サーバを運用するシステムのセキュリティ要件を定義します。TURN サーバは制限の厳しいネットワーク環境のメディア中継を支援しますが、設定を誤るとリスクをもたらす可能性があります。これらのコントロールは安全なアドレスフィルタリングと、リソース枯渇に対する保護に重点を置いています。

| # | 説明 | レベル |
| --- | --- | --- |
| **17.1.1** | Traversal Using Relays around NAT (TURN) サービスは特別な目的 (内部ネットワーク、ブロードキャスト、ループバックなど) のために予約されていない IP アドレスへのアクセスのみを許可している。これは IPv4 と IPv6 の両方のアドレスに適用することに注意する。 | 2 |
| **17.1.2** | 正規のユーザが TURN サーバ上で多数のポートを開こうとした際に、Traversal Using Relays around NAT (TURN) サービスはリソース枯渇の影響を受けない。 | 3 |

## V17.2 メディア

これらの要件は、選択転送ユニット (Selective Forwarding Unit, SFU)、多地点制御ユニット (Multipoint Control Unit, MCU)、レコーディングサーバ、ゲートウェイサーバなど、独自の WebRTC メディアサーバをホストするシステムにのみ適用します。メディアサーバはメディアストリームを処理して配信するため、ピア間の通信を保護するためにセキュリティが重要になります。WebRTC アプリケーションでは、ユーザのプライバシーと通信品質を損なう可能性のある、盗聴、改竄、サービス拒否攻撃を防ぐために、メディアストリームの保護が非常に重要です。

特に、レート制限、タイムスタンプの検証、同期クロックを使用したリアルタイムインターバルの一致、オーバーフローを防いで適切なタイミングを維持するためのバッファ管理など、フラッド攻撃に対する保護を実装する必要があります。特定のメディアセッションのパケットがあまりにも早く到着する場合、過剰なパケットをドロップすべきです。入力バリデーションを実装し、整数オーバーフローを安全に処理し、バッファオーバーフローを防止し、その他の堅牢なエラー処理技法を採用することで、不正なパケットからシステムを保護することも重要です。

中間メディアサーバを介さずに、Web ブラウザ間のピアツーピアメディア通信のみに依存するシステムは、これらの特定のメディア関連セキュリティ要件から除外されます。

このセクションでは、WebRTC のコンテキストにおけるデータグラムトランスポート層セキュリティ (Datagram Transport Layer Security, DTLS) の使用について説明します。暗号鍵の管理に対する文書化されたポリシーを持つことに関する要件は「暗号化」の章に記載されています。承認済み暗号方式に関する情報は、ASVS の暗号化の付録や、NIST SP 800‑52 Rev. 2 や BSI TR‑02102‑2 (Version 2025‑01) などのドキュメントに記載されています。

| # | 説明 | レベル |
| --- | --- | --- |
| **17.2.1** | データグラムトランスポート層セキュリティ (Datagram Transport Layer Security, DTLS) 証明書の鍵は、暗号鍵の管理に関する文書化されたポリシーに基づいて管理および保護されている。 | 2 |
| **17.2.2** | メディアサーバは、承認されたデータグラムトランスポート層セキュリティ (Datagram Transport Layer Security, DTLS) 暗号スイートと、セキュアリアルタイムトランスポートプロトコル (Secure Real-time Transport Protocol) の鍵を確立するための DTLS 拡張 (DTLS-SRTP) 用の安全な保護プロファイルを使用およびサポートするように構成されている。 | 2 |
| **17.2.3** | セキュアリアルタイムトランスポートプロトコル (Secure Real-time Transport Protocol, SRTP) 認証はメディアサーバで確認されており、リアルタイムトランスポートプロトコル (Real-time Transport Protocol, RTP) インジェクション攻撃によるサービス拒否状態や、メディアストリームへの音声または映像メディアの挿入を防止している。 | 2 |
| **17.2.4** | 不正なセキュアリアルタイムトランスポートプロトコル (Secure Real-time Transport Protocol, SRTP) パケットに遭遇した場合でも、メディアサーバは受信メディアトラフィックの処理を継続できる。 | 2 |
| **17.2.5** | 正規のユーザからセキュアリアルタイムトランスポートプロトコル (Secure Real-time Transport Protocol, SRTP) パケットが大量送信されている間も、メディアサーバは受信メディアトラフィックの処理を継続できる。 | 3 |
| **17.2.6** | メディアサーバはデータグラムトランスポート層セキュリティ (Datagram Transport Layer Security, DTLS) での "ClientHello" Race Condition 脆弱性の影響を受けておらず、メディアサーバが脆弱性を持つことを公表しているかどうかを確認し、競合状態テストを実行している。 | 3 |
| **17.2.7** | 正規のユーザからセキュアリアルタイムトランスポートプロトコル (Secure Real-time Transport Protocol, SRTP) パケットが大量送信されている間も、メディアサーバに関連付けられた音声や映像のレコーディングメカニズムは受信メディアトラフィックの処理を継続できる。 | 3 |
| **17.2.8** | データグラムトランスポート層セキュリティ (Datagram Transport Layer Security, DTLS) 証明書はセキュアリアルタイムトランスポートプロトコル (Secure Real-time Transport Protocol, SRTP) フィンガープリント属性と照合され、チェックに失敗した場合はメディアストリームを終了して、メディアストリームの真正性を確保している。 | 3 |

## V17.3 シグナリング

このセクションでは、独自の WebRTC シグナリングサーバを運用するシステムの要件を定義します。シグナリングはピアツーピア通信を調整するものであり、セッションの確立や制御を妨害する可能性のある攻撃に対して耐性を持つ必要があります。

安全なシグナリングを確保するには、システムは不正な入力を適切に処理して、負荷がかかっても利用可能であり続ける必要があります。

| # | 説明 | レベル |
| --- | --- | --- |
| **17.3.1** | フラッド攻撃中でも、シグナリングサーバは正当な受信シグナリングメッセージの処理を継続できる。これはシグナリングレベルでレート制限を実装することで実現できる。 | 2 |
| **17.3.2** | サービス拒否状態を引き起こす可能性のある不正なシグナリングメッセージに遭遇した場合でも、シグナリングサーバは正当なシグナリングメッセージの処理を継続できる。これには、入力バリデーションの実装、整数オーバーフローの安全な処理、バッファオーバーフローの防止、その他の堅牢なエラー処理技法の採用などを含む。 | 2 |

## 参考情報

詳しくは以下の情報を参照してください。

* WebRTC DTLS ClientHello DoS については [セキュリティ専門家を対象とした Enable Security のブログ投稿](https://www.enablesecurity.com/blog/novel-dos-vulnerability-affecting-webrtc-media-servers/) と、関連する [WebRTC 開発者を対象としたホワイトペーパー](https://www.enablesecurity.com/blog/webrtc-hello-race-conditions-paper/) に詳しく記載されています。
* [RFC 3550 - RTP: A Transport Protocol for Real-Time Applications](https://www.rfc-editor.org/rfc/rfc3550)
* [RFC 3711 - The Secure Real-time Transport Protocol (SRTP)](https://datatracker.ietf.org/doc/html/rfc3711)
* [RFC 5764 - Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP))](https://datatracker.ietf.org/doc/html/rfc5764)
* [RFC 8825 - Overview: Real-Time Protocols for Browser-Based Applications](https://www.rfc-editor.org/info/rfc8825)
* [RFC 8826 - Security Considerations for WebRTC](https://www.rfc-editor.org/info/rfc8826)
* [RFC 8827 - WebRTC Security Architecture](https://www.rfc-editor.org/info/rfc8827)
* [DTLS-SRTP Protection Profiles](https://www.iana.org/assignments/srtp-protection/srtp-protection.xhtml)

# 付録 A: 用語集

* **絶対最大セッション有効期間 (Absolute Maximum Session Lifetime)** - NIST では「全体タイムアウト (Overall Timeout)」とも呼ばれ、ユーザ操作に関係なく、認証後にセッションがアクティブなままでいられる最大時間です。これはセッション有効期限の構成要素です。
* **許可リスト (Allowlist)** - 許可されているデータまたは操作のリストです。例えば、入力バリデーションで許可されている文字のリストなど。
* **フォージェリ防止トークン (Anti-forgery token)** - 一つ以上のトークンがリクエストに渡され、アプリケーションサーバによって検証され、リクエストが期待したエンドポイントから来たものであることを確認するメカニズムです。
* **アプリケーションセキュリティ (Application Security)** - 基礎となるオペレーティングシステムや接続されるネットワークにフォーカスするのではなく、開放型システム間相互接続参照モデル (OSIモデル) のアプリケーション層を構成するコンポーネントの分析にフォーカスするアプリケーションレベルのセキュリティです。
* **アプリケーションセキュリティ検証 (Application Security Verification)** - OWASP ASVS に沿って実施するアプリケーションの技術的評価です。
* **アプリケーションセキュリティ検証報告書 (Application Security Verification Report)** - 特定のアプリケーションに対して検証者が作成した全体的な結果と裏付けとなる分析をまとめた報告書です。
* **認証 (Authentication)** - アプリケーションユーザが主張するアイデンティティを検証します。
* **自動検証 (Automated Verification)** - 脆弱性シグネチャを使用して問題を見つける自動ツール (動的解析ツール、静的解析ツール、その両方) を用います。
* **ブラックボックステスト (Black box testing)** - アプリケーションの機能をその内部構造や動作を調べることなく検査するソフトウェアテストの手法です。
* **共通脆弱性タイプ一覧 (Common Weakness Enumeration)** (CWE) - コミュニティが開発した一般的なソフトウェアセキュリティ脆弱性のリストです。これは共通言語、ソフトウェアセキュリティツールの測定基準、および脆弱性の特定、緩和、予防の取り組みのベースラインとして機能します。
* **コンポーネント (Component)** - 自己完結型のコード単位です。ディスクや他のコンポーネントと通信するネットワークインタフェースとの関連を持ちます。
* **クレデンシャルサービスプロバイダ (Credential Service Provider)** (CSP) - アイデンティティプロバイダ (Identity Provider, IdP) とも呼ばれます。他のアプリケーションで認証ソースとして使用される可能性のあるユーザデータのソースです。
* **クロスサイトスクリプトインクルージョン (Cross-Site Script Inclusion)** (XSSI) - クロスサイトスクリプティング (XSS) 攻撃の変種であり、Web アプリケーションが外部リソースから悪意のあるコードを取得し、そのコードを自身のコンテンツの一部として含めます。
* **クロスサイトスクリプティング (Cross-Site Scripting)** (XSS) - コンテンツにクライアントサイドスクリプトを注入することを可能にする、Web アプリケーションで典型的なセキュリティ脆弱性です。
* **暗号モジュール (Cryptographic module)** - 暗号アルゴリズムを実装し、暗号鍵を生成する、ハードウェア、ソフトウェア、ファームウェアです。
* **暗号論的に安全な疑似乱数生成器 (Cryptographically secure pseudo-random number generator)** (CSPRNG) - 暗号化での使用に適した特定を持つ疑似乱数生成器です。暗号乱数生成器 (CRNG) とも呼ばれます。
* **データグラムトランスポート層セキュリティ (Datagram Transport Layer Security)** (DTLS) – ネットワーク接続上の通信セキュリティを提供する暗号プロトコルです。TLS プロトコルをベースにしていますが、データグラム指向のプロトコル (通常は UDP 上) を保護するために改良されています。DTLS 1.3 については RFC 9147 で定義されています。
* **セキュアリアルタイムトランスポートプロトコルの鍵を確立するためのデータグラムトランスポート層セキュリティ拡張 (Datagram Transport Layer Security Extension to Establish Keys for the Secure Real-time Transport Protocol)** (DTLS-SRTP) – DTLS ハンドシェイクを使用して SRTP セッションの鍵マテリアルを確立するためのメカニズムです。RFC 5764 で定義されています。
* **設計検証 (Design Verification)** - アプリケーションのセキュリティアーキテクチャに関する技術的評価です。
* **動的アプリケーションセキュリティテスト (Dynamic Application Security Testing)** (DAST) - 実行状態にあるアプリケーションのセキュリティ脆弱性を示す状態を検出するように設計されている技法です。
* **動的検証 (Dynamic Verification)** - 脆弱性シグネチャを使用してアプリケーションの実行時に問題を見つける自動ツールを用います。
* **ファイド (Fast IDentity Online)** (FIDO) - バイオメトリクス、トラステッドプラットフォームモジュール (Trusted Platform Module, TPM)、USB セキュリティトークンなど、さまざまな認証方式を使用できるようにする一連の認証標準です。
* **ハードウェアセキュリティモジュール (Hardware Security Module)** (HSM) - 暗号鍵とその他のシークレットを保護された方法で保存するハードウェアコンポーネントです。
* **Hibernate クエリ言語 (Hibernate Query Language)** (HQL) - Hibernate ORM ライブラリで使用される SQL と似た外観のクエリ言語です。
* **HTTP Strict Transport Security** (HSTS) - 有効な証明書が提示された場合にのみ、TLS 経由でヘッダを返すドメインに接続するようにブラウザに指示するポリシーです。これは Strict-Transport-Security レスポンスヘッダフィールドを使用してアクティブ化します。
* **ハイパーテキスト転送プロトコル (HyperText Transfer Protocol)** (HTTP) - 分散、協調、ハイパーメディア情報システムのためのアプリケーションプロトコルです。World Wide Web におけるデータ通信の基盤です。
* **HyperText Transfer Protocol over SSL/TLS** (HTTPS) – トランスポート層セキュリティ (TLS) を使用して HTTP 通信を暗号化することで保護する手法です。
* **アイデンティティプロバイダ (Identity Provider)** (IdP) - NIST リファレンスではクレデンシャルサービスプロバイダ (CSP) とも呼ばれます。他のアプリケーションに認証ソースを提供するエンティティです。
* **非アクティブタイムアウト (Inactivity Timeout)** - これはユーザがアプリケーションを操作していないときにセッションがアクティブなままでいられる時間の長さです。これはセッション有効期限の構成要素です。
* **入力バリデーション (Input Validation)** - 信頼できないユーザ入力を正規化およびバリデーションします。
* **JSON Web トークン (JSON Web Token)** (JWT) - RFC 7519 は JSON データオブジェクトの標準を定義しており、オブジェクトの検証方法を説明するヘッダセクション、クレームのセットを含むボディセクション、ボディセクションのコンテンツを検証するために使用できるデジタル署名を含む署名セクションで構成されます。これは自己完結型トークンの一種です。
* **ローカルファイルインクルージョン (Local File Inclusion)** (LFI) - アプリケーションの脆弱なファイルインクルージョン手順を悪用し、サーバ上に既に存在するローカルファイルのインクルージョンにつながる攻撃です。
* **悪性コード (Malicious Code)** - アプリケーションの開発時にアプリケーションの所有者に気付かれることなく導入されるコードであり、アプリケーションの意図したセキュリティポリシーを迂回します。ウイルスやワームなどのマルウェアとは異なります！
* **マルウェア (Malware)** - アプリケーションの実行時に、ユーザや管理者に気付かれることなくアプリケーションに侵入する実行コードです。
* **メッセージ認証コード (Message authentication code)** (MAC) - MAC 生成アルゴリズムによって作成される、データの暗号チェックサムです。データの完全性と真正性を保証するために使用されます。
* **多要素認証 (Multi-factor authentication)** (MFA) - 二つ以上の単要素を含む認証です。
* **相互 TLS (Mutual TLS)** (mTLS) - TLS クライアント認証を参照してください。
* **オブジェクトリレーショナルマッピング (Object-relational Mapping)** (ORM) - アプリケーション互換オブジェクトモデルを使用して、アプリケーションプログラム内でリレーショナル/テーブルベースのデータベースを参照および照会できるようにするために使用されるシステムです。
* **ワンタイムパスワード (One-time Password)** (OTP) - 一度だけ使用するために一意に生成されるパスワードです。
* **Open Worldwide Application Security Project** (OWASP) - アプリケーションソフトウェアのセキュリティ向上に注力する、世界規模のフリーでオープンなコミュニティです。OWASP のミッションは、アプリケーションのセキュリティを「見える化」することで、人や組織がアプリケーションのセキュリティリスクについて十分な情報に基づいた決断を下せるようにすることです。[<http://www.owasp.org/>](http://www.owasp.org/) を参照してください。
* **パスワードベース鍵導出関数2 (Password-Based Key Derivation Function 2)** (PBKDF2) - 入力テキスト (パスワードなど) および追加のランダムソルト値から強力な暗号鍵を作成するために使用される特殊な一方向アルゴリズムです。元のパスワードの代わりに結果の値が保存されている場合には、パスワードをオフラインで解読することが困難になります。
* **公開鍵基盤 (Public Key Infrastructure)** (PKI) - 公開鍵をエンティティのそれぞれのアイデンティティにバインドする仕組みです。バインディングは認証局 (CA) での証明書の登録および発行のプロセスを通じて確立されます。
* **公衆交換電話網 (Public Switched Telephone Network)** (PSTN) - 固定電話と携帯電話の両方を含む従来の電話網です。
* **リアルタイムトランスポートプロトコル (Real-time Transport Protocol)** (RTP), **リアルタイムトランスポートコントロールプロトコル (Real-time Transport Control Protocol)** (RTCP) – マルチメディアストリームを転送に関連して使用される二つのプロトコルです。WebRTC スタックで使用されます。RFC 3550 で定義されています。
* **リファレンストークン (Reference Token)** - サーバに保存されている状態またはメタデータへのポインタまたは識別子として機能するトークンの一種であり、ランダムトークンや不透明トークンと呼ばれることもあります。トークン自体に関連データの一部を埋め込む自己完結型トークンとは異なり、リファレンストークンは固有の情報を含まず、代わりにコンテキストについてはサーバに依存します。リファレンストークンはセッション識別子であるか、セッション識別子を含みます。
* **依拠当事者 (Relying Party)** (RP) - 一般的には別の認証プロバイダに対して認証されたユーザに依存するアプリケーションです。アプリケーションは認証プロバイダが提供するある種のトークンまたは一連の署名済みアサーションに依存して、ユーザが本人であることを信頼します。
* **リモートファイルインクルージョン (Remote File Inclusion)** (RFI) - アプリケーションの脆弱なインクルージョン手順を悪用し、リモートファイルのインクルージョンにつながる攻撃です。
* **スケーラブルベクターグラフィックス (Scalable Vector Graphics)** (SVG) - 二次元ベースのベクターグラフィックスを記述するための XML ベースのマークアップ言語です。
* **セキュアリアルタイムトランスポートプロトコル (Secure Real-time Transport Protocol)** (SRTP), **セキュアリアルタイムトランスポートコントロールプロトコル (Secure Real-time Transport Control Protocol)** (SRTCP) - メッセージの暗号化、認証、完全性保護のサポートを提供する RTP および RTCP プロトコルのプロファイルです。RFC 3711 で定義されています。
* **セキュリティアーキテクチャ (Security Architecture)** - アプリケーションの設計を抽象化したものです。セキュリティ管理策をどこでどのように使用しているかを特定して記述します。また、ユーザデータとアプリケーションデータを保持する場所とデータの機密性について特定して記述します。
* **セキュリティアサーションマークアップ言語 (Security Assertion Markup Language)** (SAML) - ID プロバイダと依拠当事者の間で署名されたアサーション (通常は XML オブジェクト) を渡すことによって実現されるシングルサインオン認証のオープンスタンダードです。
* **セキュリティ設定 (Security Configuration)** - アプリケーションにおけるセキュリティ管理を左右する実行時設定です。
* **セキュリティ管理 (Security Control)** - セキュリティチェック (認可チェックなど) を実行する、あるいは呼び出されるとセキュリティ効果 (監査記録の生成など) をもたらす、機能やコンポーネントです。
* **セキュリティ情報およびイベント管理 (Security information and event management)** (SIEM) - 組織の IT インフラストラクチャ内のさまざまなソースからセキュリティ関連データの収集および分析することにより、脅威検出、コンプライアンス、セキュリティインシデント管理を行うシステムです。
* **自己完結型トークン (Self-Contained Token)** - サーバサイドの状態やその他の外部ストレージに依存しない一つ以上の属性をカプセル化するトークンです。これらのトークンは含まれる属性の真正性と完全性を確保し、システム間で安全な「ステートレス」な情報交換を可能にします。自己完結型トークンは一般的にデジタル署名やメッセージ認証コード (MAC) などの暗号技法を使用して保護され、データの真正性、完全性、場合によっては機密性を確保します。一般的な例としては SAML アサーションや JWT があります。
* **サーバサイドリクエストフォージェリ (Server-side Request Forgery)** (SSRF) - サーバの機能を悪用して内部リソースを読み取りまたは更新する攻撃です。攻撃者はサーバで実行されるコードがデータを読み取りまたは送信する URL を提供または変更します。
* **Session Description Protocol** (SDP) – マルチメディアセッションを設定するためのメッセージフォーマットです (WebRTC などで使用されます)。RFC 4566 で定義されています。
* **セッション識別子 (Session Identifier)**, **セッション ID (Session ID)** - バックエンドに保存されているステートフルセッションを識別する鍵です。「参照トークン」として、または「参照トークン」に含まれて、クライアントとの間で送受信されます。
* **セッショントークン (Session Token)** - この標準で使用される「包括的な」フレーズであり、ステートレスセッションメカニズム (自己完結型トークンを使用) またはステートフルセッションメカニズム (リファレンストークンを使用) で使用されるトークンや値を参照します。
* **Session Traversal Utilities for NAT** (STUN) – ピアツーピア通信を確立するために NAT トラバーサルを支援するために使用されるプロトコルです。RFC 3489 で定義されています。
* **単要素オーセンティケータ (Single-factor authenticator)** - ユーザが認証されることをチェックするメカニズムです。知識認証 (something you know) (記憶された秘密、パスワード、パスフレーズ、PIN)、生体認証 (something you are) (生体情報、指紋、顔スキャン)、または所有物認証 (something you have) (OTP トークン、スマートカードなどの暗号デバイス) のいずれかである必要があります。
* **シングルサインオン認証 (Single Sign-on Authentication)** (SSO) - ユーザがひとつのアプリケーションにログインした後に、再認証を必要とせずに自動的に他のアプリケーションにログインするものです。例えば、Google にログインすると、ユーザーは YouTube、Google Docs、Gmail などの他の Google サービスに自動的にログインします。
* **ソフトウェア部品表 (Software bill of materials)** (SBOM) - ソフトウェアアプリケーションの構築または組み立てに必要なすべてのコンポーネント、モジュール、ライブラリ、フレームワーク、その他のリソースの構造化された包括的なリストです。
* **ソフトウェア構成解析 (Software Composition Analysis)** (SCA) - 使用している特定バージョンのコンポーネントのセキュリティ脆弱性について、アプリケーションの構成、依存関係、ライブラリ、パッケージを解析するように設計された一連のテクノロジです。これは、現在一般的に SAST と呼ばれているソースコード解析と混同しないでください。
* **ソフトウェア開発ライフサイクル (Software development lifecycle)** (SDLC) - 初期要件からデプロイメントおよび保守に至るまでのソフトウェア開発における段階的なプロセスです。
* **SQL インジェクション (SQL Injection)** (SQLi) - データ駆動型アプリケーションに対する攻撃に使用されるコードインジェクション技法です。悪意のある SQL 文がエントリポイントに挿入されます。
* **ステートフルセッションメカニズム (Stateful Session Mechanism)** - ステートフルセッションメカニズムでは、アプリケーションはバックエンドでセッション状態を保持します。これは、一般的に、暗号論的に安全な疑似乱数生成 (CSPRNG) を使用して生成され、エンドユーザに発行されるセッショントークンに対応します。
* **ステートレスセッションメカニズム (Stateless Session Mechanism)** - ステートレスセッションメカニズムはクライアントに渡される自己完結型トークンを使用します。このトークンには、トークンを受け取って検証するサービス内には必ずしも保存されるわけではないセッション情報を含みます。実際には、必要なセキュリティコントロールを実施できるようにするために、サービスは一部のセッション情報 (JWT 失効リストなど) にアクセスする必要があります。
* **静的アプリケーションセキュリティテスト (Static application security testing)** (SAST) - アプリケーションコード、バイトコード、バイナリを解析し、セキュリティ脆弱性を示すコーディングや設計条件を調べるために設計された一連のテクノロジです。SAST ソリューションはアプリケーションを非実行状態で「内側から外側まで」解析します。
* **脅威モデリング (Threat Modeling)** - 脅威エージェント、セキュリティゾーン、セキュリティ管理、重要な技術資産やビジネス資産を明らかにするための、精緻なセキュリティアーキテクチャの構築に基づく手法です。
* **チェック時間と使用時間 (Time-of-check to time-of-use)** (TOCTOU) - アプリケーションがリソースを使用する前にそのリソースの状態をチェックしますが、リソースの状態はチェックと使用の間に変化する可能性がある状況です。これによりチェック結果が無効になり、この状態の不一致のためアプリケーションが無効なアクションを実行する状況を引き起こす可能性があります。
* **タイムベース OTP (Time-based OTP)** (TOTP) - OTP を生成する手法で、現在の時刻がパスワードを生成するアルゴリズムの一部として機能します。
* **TLS クライアント認証 (TLS client authentication)**, 別名 **相互 TLS (Mutual TLS)** (mTLS) - 標準的な TLS 接続では、クライアントはサーバから提供された証明書を使用してサーバのアイデンティティを検証できます。TLS クライアント認証を使用する場合、クライアントは独自の秘密鍵と証明書も使用して、サーバもクライアントのアイデンティティを検証できるようにします。
* **トランスポート層セキュリティ (Transport Layer Security)** (TLS) - ネットワークの通信セキュリティを提供する暗号プロトコルです。
* **Traversal Using Relays around NAT** (TURN) – 直接ピアツーピア通信を確立できない場合に TURN サーバをリレーとして使用する STUN プロトコルの拡張です。RFC 8656 で定義されています。
* **高信頼実行環境 (Trusted execution environment)** (TEE) - システムの他の部分に関係なく、アプリケーションを安全に実行できる分離された処理環境です。
* **Trusted Platform Module** (TPM) - 通常はマザーボードなどのより大きなハードウェアコンポーネントに接続され、そのシステムの「信頼の基点 (Root of Trust)」として機能する HSM の一種です。
* **信頼できるサービス層 (Trusted Service Layer)** - マイクロサービス、サーバレス API、サーバサイド、セキュアブートを備えたクライアントデバイス上の信頼できる API、パートナー API や外部 API など、信頼できるコントロール適用ポイントです。信頼できるとは、信頼できないユーザがその層や、その層で実装されたコントロールをバイパスやスキップできるという懸念がないことを意味します。
* **Uniform Resource Identifier** (URI)- Web ページ、メールアドレス、場所などのリソースを識別するための一意の文字列です。
* **Uniform Resource Locator** (URL) - インターネット上のリソースの場所を指定する文字列です。
* **ユニバーサル一意識別子 (Universally Unique Identifier)** (UUID) - ソフトウェアで識別子として使用される一意の照会番号です。
* **検証者 (Verifier)** - OWASP ASVS の要件に照らし合わせてアプリケーションをレビューする個人またはチームです。
* **Web Real-Time Communication** (WebRTC) – Web アプリケーション (通常はテレビ会議) でのマルチメディアストリームの転送に使用されるプロトコルスタックと関連する Web API です。SRTP, SRTCP, DTLS, SDP, STUN/TURN に基づいています。
* **WebSocket over TLS** (WSS) – TLS プロトコル上に WebSocket を階層化することで WebSocket 通信を保護する方法です。
* **What You See Is What You Get** (WYSIWYG) - レンダリングを制御するために使用されるコーディングを表示するのではなく、レンダリング時にコンテンツが実際にどのように見えるかを示すリッチコンテンツエディタのタイプです。
* **X.509 証明書 (X.509 Certificate)** - X.509 証明書は、広く受け入れられている国際的な X.509 公開鍵基盤 (PKI) 標準を使用するデジタル証明書であり、公開鍵が証明書に含まれるユーザ、コンピュータ、サービスのアイデンティティに属していることを検証します。
* **XML 外部エンティティ (XML eXternal Entity)** (XXE) - 宣言されたシステム識別子を介してローカルまたはリモートコンテンツにアクセスできる XML エンティティのタイプです。これはさまざまなインジェクション攻撃につながる可能性があります。

# 付録 B: 参考情報

以下の OWASP プロジェクトはこの標準のユーザや採用者に役立つでしょう。

## OWASP 主要プロジェクト

1. OWASP Top 10 Project: [<https://owasp.org/www-project-top-ten/>](https://owasp.org/www-project-top-ten/)
2. OWASP Web Security Testing Guide: [<https://owasp.org/www-project-web-security-testing-guide/>](https://owasp.org/www-project-web-security-testing-guide/)
3. OWASP Proactive Controls: [<https://owasp.org/www-project-proactive-controls/>](https://owasp.org/www-project-proactive-controls/)
4. OWASP Software Assurance Maturity Model (SAMM): [<https://owasp.org/www-project-samm/>](https://owasp.org/www-project-samm/)
5. OWASP Secure Headers Project: [<https://owasp.org/www-project-secure-headers/>](https://owasp.org/www-project-secure-headers/)

## OWASP チートシートシリーズプロジェクト

[このプロジェクト](https://owasp.org/www-project-cheat-sheets/) には ASVS のさまざまなトピックに関連するいくつかのチートシートがあります。

ここには ASVS へのマッピングもあります: [<https://cheatsheetseries.owasp.org/IndexASVS.html>](https://cheatsheetseries.owasp.org/IndexASVS.html)

## モバイルセキュリティ関連プロジェクト

1. OWASP Mobile Security Project: [<https://owasp.org/www-project-mobile-security/>](https://owasp.org/www-project-mobile-security/)
2. OWASP Mobile Top 10 Risks: [<https://owasp.org/www-project-mobile-top-10/>](https://owasp.org/www-project-mobile-top-10/)
3. OWASP Mobile Security Testing Guide and Mobile Application Security Verification Standard: [<https://owasp.org/www-project-mobile-security-testing-guide/>](https://owasp.org/www-project-mobile-security-testing-guide/)

## OWASP Internet of Things 関連プロジェクト

1. OWASP Internet of Things Project: [<https://owasp.org/www-project-internet-of-things/>](https://owasp.org/www-project-internet-of-things/)

## OWASP Serverless プロジェクト

1. OWASP Serverless Project: [<https://owasp.org/www-project-serverless-top-10/>](https://owasp.org/www-project-serverless-top-10/)

## その他

同様に、以下の Web サイトはこの標準のユーザや採用者に役立つでしょう。

1. SecLists Github: [<https://github.com/danielmiessler/SecLists>](https://github.com/danielmiessler/SecLists)
2. MITRE Common Weakness Enumeration: [<https://cwe.mitre.org/>](https://cwe.mitre.org/)
3. PCI Security Standards Council: [<https://www.pcisecuritystandards.org/>](https://www.pcisecuritystandards.org/)
4. PCI Data Security Standard (DSS) v3.2.1 Requirements and Security Assessment Procedures: [<https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf>](https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf)
5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures: [<https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf>](https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf)
6. PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures: [<https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf>](https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf)
7. OWASP ASVS 4.0 Testing Guide [<https://github.com/BlazingWind/OWASP-ASVS-4.0-testing-guide>](https://github.com/BlazingWind/OWASP-ASVS-4.0-testing-guide)

# 付録 C: 暗号化標準

「暗号化」の章は単にベストプラクティスを定義するだけではありません。暗号の原則に対する理解を深め、より耐性のある最新のセキュリティ手法の採用を促進することを目的としています。この付録では、「暗号化」の章で概説されている包括的な標準を保管するために、各要件に関する詳細な技術情報を提供します。

この付録ではさまざまな暗号メカニズムの承認レベルを定義します。

* 承認済み (Approved) (A) メカニズムはアプリケーションで使用できます。
* レガシー (Legacy) メカニズム (L) はアプリケーションで使用すべきではありませんが、既存のレガシーアプリケーションやコードとの互換性のためだけに依然として使用されるかもしれません。このようなメカニズムの使用自体は現在のところ脆弱性とはみなされませんが、できるだけ早く、より安全で将来性のあるメカニズムに置き換えるべきです。
* 不許可 (Disallowed) メカニズム (D) は、現時点で破壊されているとみなされているか、十分なセキュリティを提供していないため、使用してはいけません。

このリストは、以下のようなさまざまな理由により、特定のアプリケーションのコンテキストにおいて上書きされる可能性があります。

* 暗号分野における新たな進化
* 規制への準拠

## 暗号インベントリとドキュメント

このセクションでは V11.1 暗号インベントリとドキュメントに対する追加情報を提供します。

アルゴリズム、鍵、証明書などの暗号資産を定期的に発見し、インベントリ化し、評価することが重要です。レベル 3 では、これにはアプリケーションでの暗号の使用を発見するための静的スキャンおよび動的スキャンの使用を含むべきです。SAST や DAST などのツールがこれに役立つかもしれませんが、より包括的なカバレッジを得るには専用のツールが必要になるかもしれません。フリーウェアのツールの例には以下のものがあります。

* [CryptoMon - Network Cryptography Monitor - using eBPF, written in python](https://github.com/Santandersecurityresearch/CryptoMon)
* [Cryptobom Forge Tool: Generating Comprehensive CBOMs from CodeQL Outputs](https://github.com/Santandersecurityresearch/cryptobom-forge)

## 暗号パラメータの等価な強度

さまざまな暗号システムの相対的なセキュリティ強度は以下の表のとおりです ([NIST SP 800-57 Part 1](https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final), p.71 より):

| セキュリティ強度 | 対称鍵アルゴリズム | 有限体 | 整数因数分解 | 楕円曲線 |
| --- | --- | --- | --- | --- |
| <= 80 | 2TDEA | L = 1024 N = 160 | k = 1024 | f = 160-223 |
| 112 | 3TDEA | L = 2048 N = 224 | k = 2048 | f = 224-255 |
| 128 | AES-128 | L = 3072 N = 256 | k = 3072 | f = 256-383 |
| 192 | AES-192 | L = 7680 N = 384 | k = 7680 | f = 384-511 |
| 256 | AES-256 | L = 15360 N = 512 | k = 15360 | f = 512+ |

適用例:

* 有限体暗号: DSA, FFDH, MQV
* 整数因数分解暗号: RSA
* 楕円曲線暗号: ECDSA, EdDSA, ECDH, MQV

注: このセクションでは量子コンピュータが存在しないことを前提としています。そのようなコンピュータが存在する場合、最後の 3 列の推定値はもはや有効ではなくなります。

## 乱数値

このセクションでは V11.5 乱数値に対する追加情報を提供します。

| 名称 | バージョン/リファレンス | 備考 | ステータス |
| --- | --- | --- | --- |
| /dev/random | Linux 4.8 以降 [(2016 年 10 月)](https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=818e607b57c94ade9824dad63a96c2ea6b21baf3)、iOS、Android、その他の Linux ベースの POSIX オペレーティングシステムにもあります。[RFC7539](https://datatracker.ietf.org/doc/html/rfc7539) に基づいています。 | ChaCha20 ストリームを利用します。iOS [SecRandomCopyBytes](https://developer.apple.com/documentation/security/secrandomcopybytes(_:_:_:)?language=objc) および Android [Secure Random](https://developer.android.com/reference/java/security/SecureRandom) にあり、それぞれに正しい設定が提供されています。 | A |
| /dev/urandom | ランダムデータを提供するための Linux カーネルの特殊ファイルです。 | ハードウェアのランダム性から高性能のエントロピーソースを提供します。 | A |
| AES-CTR-DRBG | [NIST SP800-90A](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf) | [BCRYPT\_RNG\_ALGORITHM](https://learn.microsoft.com/en-us/windows/win32/seccng/cng-algorithm-identifiers) で設定された [Windows CNG API BCryptGenRandom](https://learn.microsoft.com/en-us/windows/win32/api/bcrypt/nf-bcrypt-bcryptgenrandom) などの一般的な実装で使用されます。 | A |
| HMAC-DRBG | [NIST SP800-90A](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf) |  | A |
| Hash-DRBG | [NIST SP800-90A](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf) |  | A |
| getentropy() | [OpenBSD](https://man.openbsd.org/getentropy.2)、[Linux glibc 2.25 以降](https://man7.org/linux/man-pages/man3/getentropy.3.html)、[macOS 10.12 以降](https://support.apple.com/en-gb/guide/security/seca0c73a75b/web) で利用可能です。 | シンプルで最小限の API で、カーネルのエントロピーソースから安全なランダムバイトを直接提供します。より現代的であり、古い API に関連する落とし穴を回避します。 | A |

HMAC-DRBG または Hash-DRBG で使用される基礎となるハッシュ関数はこの用途に対して承認されていなければなりません。

## 暗号アルゴリズム

このセクションでは V11.3 暗号アルゴリズムに対する追加情報を提供します。

承認済み暗号アルゴリズムを優先順に並べています。

| 対称鍵アルゴリズム | リファレンス | ステータス |
| --- | --- | --- |
| AES-256 | [FIPS 197](https://csrc.nist.gov/pubs/fips/197/final) | A |
| Salsa20 | [Salsa 20 specification](https://cr.yp.to/snuffle/spec.pdf) | A |
| XChaCha20 | [XChaCha20 Draft](https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-xchacha-03) | A |
| XSalsa20 | [Extending the Salsa20 nonce](https://cr.yp.to/snuffle/xsalsa-20110204.pdf) | A |
| ChaCha20 | [RFC 8439](https://www.rfc-editor.org/info/rfc8439) | A |
| AES-192 | [FIPS 197](https://csrc.nist.gov/pubs/fips/197/final) | A |
| AES-128 | [FIPS 197](https://csrc.nist.gov/pubs/fips/197/final) | L |
| 2TDEA |  | D |
| TDEA (3DES/3DEA) |  | D |
| IDEA |  | D |
| RC4 |  | D |
| Blowfish |  | D |
| ARC4 |  | D |
| DES |  | D |

### AES 暗号モード

AES などのブロック暗号はさまざまな動作モードで使用できます。電子コードブック (ECB) などの多くの動作モードは安全ではないため、使用してはいけません。ガロア/カウンタモード (Galois/Counter Mode, GCM) とカウンタ暗号ブロック連鎖メッセージ認証コード (CCM) の動作モードは、認証済み暗号化を提供するため、現代のアプリケーションでは使用すべきです。

承認済みモードを優先順に並べています。

| モード | 認証済み | リファレンス | ステータス | 制限 |
| --- | --- | --- | --- | --- |
| GCM | Yes | [NIST SP 800-38D](https://csrc.nist.gov/pubs/sp/800/38/d/final) | A |  |
| CCM | Yes | [NIST SP 800-38C](https://csrc.nist.gov/pubs/sp/800/38/c/upd1/final) | A |  |
| CBC | No | [NIST SP 800-38A](https://csrc.nist.gov/pubs/sp/800/38/a/final) | L |  |
| CCM-8 | Yes |  | D |  |
| ECB | No |  | D |  |
| CFB | No |  | D |  |
| OFB | No |  | D |  |
| CTR | No |  | D |  |

注:

* すべての暗号化されたメッセージは認証されなければなりません。CBC モードを使用する場合は必ず、メッセージを検証するために、関連するハッシュ MAC アルゴリズムがなければなりません (MUST)。一般的に、これは Encrypt-Then-Hash 方式で適用されなければなりません (MUST) (ただし、TLS 1.2 では代わりに Hash-Then-Encrypt を使用します)。これが保証できない場合、CBC を使用してはいけません (MUST NOT)。MAC アルゴリズムなしでの暗号化が許される唯一のアプリケーションはディスク暗号化です。
* CBC を使用する場合、パディングの検証が一定時間で行われることを保証する必要があります。
* CCM-8 を使用する場合、MAC タグは 64 ビットのセキュリティしか持ちません。これは、少なくとも 128 ビットのセキュリティを必要とする要件 6.2.9 に準拠していません。
* ディスク暗号化は ASVS のスコープ外と考えられます。したがって、この付録ではディスク暗号化の承認済みの方式は記載しません。この用途では、通常、認証なしの暗号化が受け入れられ、XTS、XEX、LRW モードが一般的に使用されます。

### 鍵ラッピング

暗号鍵ラップ (および対応する鍵アンラップ) は、追加の暗号メカニズムを使用して既存の鍵をカプセル化 (つまりラップ) し、転送中などに元の鍵が明らかに露出しないようにすることで、既存の鍵を保護する方法です。元の鍵を保護するために使用されるこの追加の鍵はラップ鍵と呼ばれます。

この操作は、信頼できないとみなされる場所で鍵を保護したい場合、あるいは信頼できないネットワーク上やアプリケーション内で機密鍵を送信したい場合に実行できます。 ただし、ラップ/アンラップ手順を実行する前に、元の鍵の性質 (アイデンティティや目的など) を理解することを真剣に検討すべきです。これは、セキュリティと、特に鍵の機能 (署名など) の監査証跡や適切な鍵の保存を含むコンプライアンスの点で、ソースとターゲットの両方のシステム/アプリケーションに影響を及ぼす可能性があります。

特に、鍵ラッピングには、[NIST SP 800-38F](https://csrc.nist.gov/pubs/sp/800/38/f/final) に従い、量子脅威に対する将来を見据えた対策を考慮して、AES-256 を使用しなければなりません (MUST)。AES を使用する暗号モードは優先順に以下のとおりです:

| 鍵ラッピング | リファレンス | ステータス |
| --- | --- | --- |
| KW | [NIST SP 800-38F](https://csrc.nist.gov/pubs/sp/800/38/f/final) | A |
| KWP | [NIST SP 800-38F](https://csrc.nist.gov/pubs/sp/800/38/f/final) | A |

AES-192 と AES-128 はユースケースで必要な場合に使用できます (MAY) が、その理由はエンティティの暗号インベントリに文書化されなければなりません (MUST)。

### 認証された暗号

ディスク暗号化を除いて、暗号化されたデータは何かしらの形で認証された暗号 (AE) スキーム、通常は関連データ付き認証暗号 (AEAD) スキームを使用して、認可されていない変更から保護されなければなりません。

アプリケーションは承認済みの AEAD スキームを使用することをお勧めします。代わりに、承認済みの暗号スキームと承認済みの MAC アルゴリズムを Encrypt-then-MAC 構造で組み合わせることもできます。

MAC-then-encrypt はレガシーアプリケーションとの互換性のために依然として許可されています。これは古い暗号スイートを使用する TLS v1.2 で使用されます。

| AEAD メカニズム | リファレンス | ステータス |
| --- | --- | --- |
| AES-GCM | [SP 800-38D](https://csrc.nist.gov/pubs/sp/800/38/d/final) | A |
| AES-CCM | [SP 800-38C](https://csrc.nist.gov/pubs/sp/800/38/c/upd1/final) | A |
| ChaCha-Poly1305 | [RFC 7539](https://datatracker.ietf.org/doc/html/rfc7539) | A |
| AEGIS-256 | [AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)](https://competitions.cr.yp.to/round3/aegisv11.pdf) | A |
| AEGIS-128 | [AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)](https://competitions.cr.yp.to/round3/aegisv11.pdf) | A |
| AEGIS-128L | [AEGIS: A Fast Authenticated Encryption Algorithm (v1.1)](https://competitions.cr.yp.to/round3/aegisv11.pdf) | A |
| Encrypt-then-MAC |  | A |
| MAC-then-encrypt |  | L |

## ハッシュ関数

このセクションでは V11.4 ハッシュ化とハッシュベース関数に対する追加情報を提供します。

### 一般的なユースケースでのハッシュ関数

以下の表は、デジタル署名などの一般的な暗号ユースケースで承認されているハッシュ関数を示しています。

* 承認済みハッシュ関数は強力な衝突耐性を備えており、高セキュリティのアプリケーションに適しています。
* これらのアルゴリズムの中には、適切な暗号鍵管理と併用すると攻撃に対する強力な耐性を発揮するものがあるため、HMAC、KDF、RBG 機能についても追加的に承認されています。
* 出力が 254 ビット未満のハッシュ関数は衝突耐性が不十分であるため、デジタル署名や衝突耐性を必要とするその他の用途には使用してはいけません。その他の用途では、レガシーシステムとの互換性と検証にのみ使用できます (ONLY) が、新しい設計には使用してはいけません。

| ハッシュ関数 | リファレンス | ステータス | 制限 |
| --- | --- | --- | --- |
| SHA3-512 | [FIPS 202](https://csrc.nist.gov/pubs/fips/202/final) | A |  |
| SHA-512 | [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | A |  |
| SHA3-384 | [FIPS 202](https://csrc.nist.gov/pubs/fips/202/final) | A |  |
| SHA-384 | [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | A |  |
| SHA3-256 | [FIPS 202](https://csrc.nist.gov/pubs/fips/202/final) | A |  |
| SHA-512/256 | [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | A |  |
| SHA-256 | [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | A |  |
| SHAKE256 | [FIPS 202](https://csrc.nist.gov/pubs/fips/202/final) | A |  |
| BLAKE2s | [BLAKE2: simpler, smaller, fast as MD5](https://eprint.iacr.org/2013/322) | A |  |
| BLAKE2b | [BLAKE2: simpler, smaller, fast as MD5](https://eprint.iacr.org/2013/322) | A |  |
| BLAKE3 | [BLAKE3 one function, fast everywhere](https://github.com/BLAKE3-team/BLAKE3-specs/raw/master/blake3.pdf) | A |  |
| SHA-224 | [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | L | Not suitable for HMAC, KDF, RBG, digital signatures |
| SHA-512/224 | [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | L | Not suitable for HMAC, KDF, RBG, digital signatures |
| SHA3-224 | [FIPS 202](https://csrc.nist.gov/pubs/fips/202/final) | L | Not suitable for HMAC, KDF, RBG, digital signatures |
| SHA-1 | [RFC 3174](https://www.rfc-editor.org/info/rfc3174) & [RFC 6194](https://www.rfc-editor.org/info/rfc6194) | L | Not suitable for HMAC, KDF, RBG, digital signatures |
| CRC (any length) |  | D |  |
| MD4 | [RFC 1320](https://www.rfc-editor.org/info/rfc1320) | D |  |
| MD5 | [RFC 1321](https://www.rfc-editor.org/info/rfc1321) | D |  |

### パスワード保存のためのハッシュ関数

安全なパスワードハッシュには、専用のハッシュ関数を使用しなければなりません。これらの低速ハッシュアルゴリズムは、パスワードクラッキングの計算難易度を上げることで、ブルートフォース攻撃や辞書攻撃を軽減します。

| 鍵導出関数 | リファレンス | 必要なパラメータ | ステータス |
| --- | --- | --- | --- |
| argon2id | [RFC 9106](https://www.rfc-editor.org/info/rfc9106) | t = 1: m ≥ 47104 (46 MiB), p = 1t = 2: m ≥ 19456 (19 MiB), p = 1t ≥ 3: m ≥ 12288 (12 MiB), p = 1 | A |
| scrypt | [RFC 7914](https://www.rfc-editor.org/info/rfc7914) | p = 1: N ≥ 2^17 (128 MiB), r = 8p = 2: N ≥ 2^16 (64 MiB), r = 8p ≥ 3: N ≥ 2^15 (32 MiB), r = 8 | A |
| bcrypt | [A Future-Adaptable Password Scheme](https://www.researchgate.net/publication/2519476_A_Future-Adaptable_Password_Scheme) | cost ≥ 10 | A |
| PBKDF2-HMAC-SHA-512 | [NIST SP 800-132](https://csrc.nist.gov/pubs/sp/800/132/final), [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | iterations ≥ 210,000 | A |
| PBKDF2-HMAC-SHA-256 | [NIST SP 800-132](https://csrc.nist.gov/pubs/sp/800/132/final), [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | iterations ≥ 600,000 | A |
| PBKDF2-HMAC-SHA-1 | [NIST SP 800-132](https://csrc.nist.gov/pubs/sp/800/132/final), [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | iterations ≥ 1,300,000 | L |

承認済みのパスワードベースの鍵導出関数はパスワードストレージとして使用できます。

## 鍵導出関数 (KDF)

### 汎用の鍵導出関数

| 鍵導出関数 | リファレンス | ステータス |
| --- | --- | --- |
| HKDF | [RFC 5869](https://www.rfc-editor.org/info/rfc5869) | A |
| TLS 1.2 PRF | [RFC 5248](https://www.rfc-editor.org/info/rfc5248) | L |
| MD5-based KDFs | [RFC 1321](https://www.rfc-editor.org/info/rfc1321) | D |
| SHA-1-based KDFs | [RFC 3174](https://www.rfc-editor.org/info/rfc3174) & [RFC 6194](https://www.rfc-editor.org/info/rfc6194) | D |

### パスワードベースの鍵導出関数

| 鍵導出関数 | リファレンス | 必要なパラメータ | ステータス |
| --- | --- | --- | --- |
| argon2id | [RFC 9106](https://www.rfc-editor.org/info/rfc9106) | t = 1: m ≥ 47104 (46 MiB), p = 1t = 2: m ≥ 19456 (19 MiB), p = 1t ≥ 3: m ≥ 12288 (12 MiB), p = 1 | A |
| scrypt | [RFC 7914](https://www.rfc-editor.org/info/rfc7914) | p = 1: N ≥ 2^17 (128 MiB), r = 8p = 2: N ≥ 2^16 (64 MiB), r = 8p ≥ 3: N ≥ 2^15 (32 MiB), r = 8 | A |
| PBKDF2-HMAC-SHA-512 | [NIST SP 800-132](https://csrc.nist.gov/pubs/sp/800/132/final), [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | iterations ≥ 210,000 | A |
| PBKDF2-HMAC-SHA-256 | [NIST SP 800-132](https://csrc.nist.gov/pubs/sp/800/132/final), [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | iterations ≥ 600,000 | A |
| PBKDF2-HMAC-SHA-1 | [NIST SP 800-132](https://csrc.nist.gov/pubs/sp/800/132/final), [FIPS 180-4](https://csrc.nist.gov/pubs/fips/180-4/upd1/final) | iterations ≥ 1,300,000 | L |

## 鍵交換メカニズム

このセクションでは V11.6 公開鍵暗号に対する追加情報を提供します。

### KEX スキーム

すべての鍵交換スキームに対して 112 ビット以上のセキュリティ強度が確保されていなければならず (MUST)、その実装は以下の表のパラメータ選択に従わなければなりません (MUST)。

| スキーム | ドメインパラメータ | 前方秘匿性 | ステータス |
| --- | --- | --- | --- |
| Finite Field Diffie-Hellman (FFDH) | L >= 3072 & N >= 256 | Yes | A |
| Elliptic Curve Diffie-Hellman (ECDH) | f >= 256-383 | Yes | A |
| Encrypted key transport with RSA-PKCS#1 v1.5 |  | No | D |

ここでのパラメータは以下のとおりです:

* k は RSA 鍵の鍵サイズです。
* L は有限体暗号の公開鍵 (public key) のサイズであり、N は秘密鍵 (private key) のサイズです。
* f は ECC の鍵サイズの範囲です。

新しい実装では [NIST SP 800-56A](https://csrc.nist.gov/pubs/sp/800/56/a/r3/final) & [B](https://csrc.nist.gov/pubs/sp/800/56/b/r2/final) および [NIST SP 800-77](https://csrc.nist.gov/pubs/sp/800/77/r1/final) に準拠していないスキームを使用してはいけません (MUST NOT)。特に IKEv1 は本番環境で使用してはいけません (MUST NOT)。

### Diffie-Hellman グループ

Diffie-Hellman 鍵交換の実装には以下のグループが承認されています。セキュリティ強度は [NIST SP 800-56A](https://csrc.nist.gov/pubs/sp/800/56/a/r3/final), Appendix D および [NIST SP 800-57 Part 1 Rev.5](https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final) に記載されています。

| グループ | ステータス |
| --- | --- |
| P-224, secp224r1 | A |
| P-256, secp256r1 | A |
| P-384, secp384r1 | A |
| P-521, secp521r1 | A |
| K-233, sect233k1 | A |
| K-283, sect283k1 | A |
| K-409, sect409k1 | A |
| K-571, sect571k1 | A |
| B-233, sect233r1 | A |
| B-283, sect283r1 | A |
| B-409, sect409r1 | A |
| B-571, sect571r1 | A |
| Curve448 | A |
| Curve25519 | A |
| MODP-2048 | A |
| MODP-3072 | A |
| MODP-4096 | A |
| MODP-6144 | A |
| MODP-8192 | A |
| ffdhe2048 | A |
| ffdhe3072 | A |
| ffdhe4096 | A |
| ffdhe6144 | A |
| ffdhe8192 | A |

## メッセージ認証コード (MAC)

メッセージ認証コード (MAC) は、メッセージの完全性と真正性を検証するために使用される暗号構造です。MAC は、メッセージと共有鍵 (secret key) を入力として受け取り、固定サイズのタグ (MAC 値) を生成します。MAC は安全な通信プロトコル (TLS/SSL など) で広く使用され、パーティ間で交換されるメッセージが本物で無傷であることを確保します。

| MAC アルゴリズム | リファレンス | ステータス | 制限 |
| --- | --- | --- | --- |
| HMAC-SHA-256 | [RFC 2104](https://www.rfc-editor.org/info/rfc2104) & [FIPS 198-1](https://csrc.nist.gov/pubs/fips/198-1/final) | A |  |
| HMAC-SHA-384 | [RFC 2104](https://www.rfc-editor.org/info/rfc2104) & [FIPS 198-1](https://csrc.nist.gov/pubs/fips/198-1/final) | A |  |
| HMAC-SHA-512 | [RFC 2104](https://www.rfc-editor.org/info/rfc2104) & [FIPS 198-1](https://csrc.nist.gov/pubs/fips/198-1/final) | A |  |
| KMAC128 | [NIST SP 800-185](https://csrc.nist.gov/pubs/sp/800/185/final) | A |  |
| KMAC256 | [NIST SP 800-185](https://csrc.nist.gov/pubs/sp/800/185/final) | A |  |
| BLAKE3 (keyed\_hash mode) | [BLAKE3 one function, fast everywhere](https://github.com/BLAKE3-team/BLAKE3-specs/raw/master/blake3.pdf) | A |  |
| AES-CMAC | [RFC 4493](https://datatracker.ietf.org/doc/html/rfc4493) & [NIST SP 800-38B](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf) | A |  |
| AES-GMAC | [NIST SP 800-38D](https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf) | A |  |
| Poly1305-AES | [The Poly1305-AES message-authentication code](https://cr.yp.to/mac/poly1305-20050329.pdf) | A |  |
| HMAC-SHA-1 | [RFC 2104](https://www.rfc-editor.org/info/rfc2104) & [FIPS 198-1](https://csrc.nist.gov/pubs/fips/198-1/final) | L |  |
| HMAC-MD5 | [RFC 1321](https://www.rfc-editor.org/info/rfc1321) | D |  |

## デジタル署名

署名スキームは [NIST SP 800-57 Part 1](https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final) に従って承認された鍵サイズとパラメータを使用しなければなりません (MUST):

| 署名アルゴリズム | リファレンス | ステータス |
| --- | --- | --- |
| EdDSA (Ed25519, Ed448) | [RFC 8032](https://www.rfc-editor.org/info/rfc8032) | A |
| XEdDSA (Curve25519, Curve448) | [XEdDSA](https://signal.org/docs/specifications/xeddsa/) | A |
| ECDSA (P-256, P-384, P-521) | [FIPS 186-4](https://csrc.nist.gov/pubs/fips/186-5/final) | A |
| RSA-RSSA-PSS | [RFC 8017](https://www.rfc-editor.org/info/rfc8017) | A |
| RSA-SSA-PKCS#1 v1.5 | [RFC 8017](https://www.rfc-editor.org/info/rfc8017) | D |
| DSA (any key size) | [FIPS 186-4](https://csrc.nist.gov/pubs/fips/186-4/final) | D |

## ポスト量子暗号標準

PQC 実装は、堅牢化されたコードや実装リファレンスはまだ最低限しか存在しないため、[FIPS-203](https://csrc.nist.gov/pubs/fips/203/ipd)/[204](https://csrc.nist.gov/pubs/fips/204/ipd)/[205](https://csrc.nist.gov/pubs/fips/205/ipd) に準拠していなければなりません。 <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>

提案されている [mlkem768x25519](https://datatracker.ietf.org/doc/draft-kwiatkowski-tls-ecdhe-mlkem/03/) ポスト量子ハイブリッド TLS 鍵交換方式は、[Firefox リリース 132](https://www.mozilla.org/en-US/firefox/132.0/releasenotes/) や [Chrome リリース 131](https://security.googleblog.com/2024/09/a-new-path-for-kyber-on-web.html) などの主要なブラウザでサポートされています。暗号テスト環境や、業界や政府が承認したライブラリ内で利用可能な場合に使用できます (MAY)。

# 付録 D: 推奨事項

## はじめに

アプリケーションセキュリティ検証標準 (ASVS) のバージョン 5.0 を準備する中で、5.0 の要件に含めるべきではない既存および新規に提案された項目が多数あることが明らかになりました。これは 5.0 の定義により ASVS のスコープではないか、あるいは、良いアイデアであったとしても、必須とすることはできないと考えたためです。

これらの項目を完全に失いたくはなかったので、この付録にその一部を収めました。

## 推奨される、スコープ内のメカニズム

以下の項目は ASVS のスコープ内です。これらを必須にすべきではありませんが、セキュアアプリケーションの一部として考慮することを強くお勧めします。

* パスワード強度メーターを提供して、ユーザーがより強力なパスワードを設定できるようにする。
* アプリケーションのルートまたは .well-known ディレクトリに一般に利用可能な security.txt を作成して、セキュリティ問題について所有者に連絡するためのリンクまたは電子メールアドレスを明確に定義する。
* 信頼できるサービス層でのバリデーションに加えて、クライアントサイドの入力バリデーションを実施すべきである。これは何者かがアプリケーションを攻撃しようとしてクライアントサイドのコントロールをバイパスしたことを発見する良い機会を提供するためである。
* robots.txt ファイル、X-Robots-Tag レスポンスヘッダ、または robots HTML meta タグを使用して、間違ってアクセス可能な機密ページが検索エンジンに表示されることを防ぐ。
* GraphQL を使用する場合、認可ロジックを GraphQL やリゾルバレイヤではなくビジネスロジックレイヤで実装して、個別のインタフェースごとに認可を処理する必要がないようにする。

参考情報:

* [RFC へのリンクを含む security.txt の詳細情報](https://securitytxt.org/)

## ソフトウェアセキュリティの原則

以下の項目は以前 ASVS にありましたが、実際には要件ではありません。むしろ、セキュリティコントロールを実装する際に考慮すべき原則であり、これに従うと、より堅牢なコントロールにつながります。以下があります。

* セキュリティコントロールは集中管理され、簡潔 (経済的設計) で、検証可能で、セキュアで、再利用可能であるべきです。これによりコントロールの重複、欠落、効果がないものを回避できます。
* 可能な限り、ゼロからコントロールを実装するのではなく、前もって作成され、十分に精査されたセキュリティコントロール実装を使用します。
* 理想的には、保護されたデータおよびリソースにアクセスするために、単一の十分に検証されたアクセス制御メカニズムを使用すべきです。コピー＆ペーストやセキュアではない代替パスを回避するために、すべてのリクエストはこの単一のメカニズムを通過すべきです。
* 属性または機能ベースのアクセス制御は推奨パターンであり、コードが単に役割ではなく機能やデータ項目に対するユーザの認可を確認しています。権限は引き続き役割を使用して割り当てるべきです。

## ソフトウェアセキュリティプロセス

ASVS 5.0 から削除されたセキュリティプロセスが多数ありますが、それはまだ良いアイデアです。これらのプロセスを効果的に実装する方法については、OWASP SAMM プロジェクトが優れた情報源となるかもしれません。以前 ASVS に含まれていた項目には以下のものがあります。

* 開発のすべての段階でセキュリティに対処するセキュアソフトウェア開発ライフサイクルを使用する。
* 脅威を特定し、対応策を計画し、適切なリスク対応を促進し、セキュリティテストをガイドするために、設計変更やスプリント計画ごとに脅威モデリングを使用する。
* すべてのユーザストーリーと機能に「ユーザとして、自分のプロフィールを閲覧および編集できる必要がある。他の人のプロフィールを閲覧または編集できないようにする必要がある」のような機能的なセキュリティ制約が含まれている。
* セキュアコーディングチェックリスト、セキュリティ要件、ガイドライン、ポリシーがすべての開発者とテスト担当者に利用可能である。
* アプリケーションのソースコードには、バックドア、悪意のあるコード (サラミ攻撃、ロジックボム、時限爆弾など)、文書化されていない機能や隠された機能 (イースターエッグ、安全でないデバッグツールなど) がないことを確認するための継続的なプロセスが存在する。このセクションに準拠するには、サードパーティライブラリを含むソースコードへの完全なアクセスなしには不可能であり、したがって、最高レベルのセキュリティを必要とするアプリケーションにのみ適していると考えられる。
* デプロイされた環境における構成のドリフトを検出し、対応するためのメカニズムが導入されている。これには、不変のインフラストラクチャ、安全なベースラインからの自動再デプロイメント、承認された構成と現在の状態を比較するドリフト検出ツールの使用を含むことがある。
* すべてのサードパーティ製品、ライブラリ、フレームワーク、サービスにおいて、それぞれの推奨事項に従って、設定の堅牢化が実行されている。

参考情報:

* [OWASP Threat Modeling Cheat Sheet](https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html)
* [OWASP Threat modeling](https://owasp.org/www-community/Application_Threat_Modeling)
* [OWASP Software Assurance Maturity Model Project](https://owasp.org/www-project-samm/)
* [Microsoft SDL](https://www.microsoft.com/en-us/securityengineering/sdl/)

# 付録 E - 貢献者

ASVS 4.0.0 のリリース以降にコメントやプルリクエストをオープンしていただいた以下の方々の貢献に深く感謝いたします。

誤りにお気づきの場合や、名前を別の形にしたい場合には、お知らせください。

|  |  |  |  |
| --- | --- | --- | --- |
| Johan Sydseter ([sydseter](https://github.com/sydseter)) | luis servin ([lfservin](https://github.com/lfservin)) | Oleksii Dovydkov ([oleksiidov](https://github.com/oleksiidov)) | IZUKA Masahiro ([maizuka](https://github.com/maizuka)) |
| James Sulinski ([jsulinski](https://github.com/jsulinski)) | Eli Saad ([ThunderSon](https://github.com/ThunderSon)) | [kkshitish9](https://github.com/kkshitish9) | Andrew van der Stock ([vanderaj](https://github.com/vanderaj)) |
| Rick M ([kingthorin](https://github.com/kingthorin)) | Bankde Eakasit ([Bankde](https://github.com/Bankde)) | Michael Gargiullo ([mgargiullo](https://github.com/mgargiullo)) | Raphael Dunant ([Racater](https://github.com/Racater)) |
| Cesar Kohl ([cesarkohl](https://github.com/cesarkohl)) | [inaz0](https://github.com/inaz0) | Joerg Bruenner ([JoergBruenner](https://github.com/JoergBruenner)) | David Deatherage ([securitydave](https://github.com/securitydave)) |
| John Carroll ([yosignals](https://github.com/yosignals)) | Jim Fenton ([jimfenton](https://github.com/jimfenton)) | Matteo Pace ([M4tteoP](https://github.com/M4tteoP)) | Sebastien gioria ([SPoint42](https://github.com/SPoint42)) |
| Steven van der Baan ([vdbaan](https://github.com/vdbaan)) | Jeremy Bonghwan Choi ([jeremychoi](https://github.com/jeremychoi)) | [craig-shony](https://github.com/craig-shony) | Riccardo Sirigu ([ricsirigu](https://github.com/ricsirigu)) |
| Tomasz Wrobel ([tw2as](https://github.com/tw2as)) | Alena Dubeshko ([belalena](https://github.com/belalena)) | Rafael Green ([RafaelGreen1](https://github.com/RafaelGreen1)) | [mjang-cobalt](https://github.com/mjang-cobalt) |
| [clallier94](https://github.com/clallier94) | Kevin W. Wall ([kwwall](https://github.com/kwwall)) | Jordan Sherman ([jsherm-fwdsec](https://github.com/jsherm-fwdsec) / [deleterepo](https://github.com/deleterepo)) | Ingo Rauner ([ingo-rauner](https://github.com/ingo-rauner)) |
| Dirk Wetter ([drwetter](https://github.com/drwetter)) | Moshe Zioni ([moshe-apiiro](https://github.com/moshe-apiiro)) | Patrick Dwyer ([coderpatros](https://github.com/coderpatros)) | David Clarke ([davidclarke-au](https://github.com/davidclarke-au)) |
| Takaharu Ogasa ([takaharuogasa](https://github.com/takaharuogasa)) | Arkadii Yakovets ([arkid15r](https://github.com/arkid15r)) | Motoyasu Saburi ([motoyasu-saburi](https://github.com/motoyasu-saburi)) | [leirn](https://github.com/leirn) |
| [wet-certitude](https://github.com/wet-certitude) | [timhemel](https://github.com/timhemel) | RL Thornton ([thornshadow99](https://github.com/thornshadow99)) | Thomas Bandt ([aspnetde](https://github.com/aspnetde)) |
| Roel Storms ([roelstorms](https://github.com/roelstorms)) | Jeroen Willemsen ([commjoen](https://github.com/commjoen)) | [anonymous-31](https://github.com/anonymous-31) | Kamran Saifullah ([deFr0ggy](https://github.com/deFr0ggy)) |
| Steve Springett ([stevespringett](https://github.com/stevespringett)) | Spyros ([northdpole](https://github.com/northdpole)) | Hans Herrera ([hansphp](https://github.com/hansphp)) | [Marx314](https://github.com/Marx314) |
| [CarlosAllendes](https://github.com/CarlosAllendes) | Yonah Russ ([yruss972](https://github.com/yruss972)) | Sander Maijers ([sanmai-NL](https://github.com/sanmai-NL)) | Luboš Bretschneider ([bretik](https://github.com/bretik)) |
| Eva Sarafianou ([esarafianou](https://github.com/esarafianou)) | [ataseren](https://github.com/ataseren) | Steve Thomas ([Sc00bz](https://github.com/Sc00bz)) | Dominique RIGHETTO ([righettod](https://github.com/righettod)) |
| Steven van der Baan ([svdb-ncc](https://github.com/svdb-ncc)) | Michael Vacarella ([Aif4thah](https://github.com/Aif4thah)) | Tonimir Kisasondi ([tkisason](https://github.com/tkisason)) | Stefan Streichsbier ([streichsbaer](https://github.com/streichsbaer)) |
| [hi-unc1e](https://github.com/hi-unc1e) | sb3k ([starbuck3000](https://github.com/starbuck3000)) | [mario-platt](https://github.com/mario-platt) | Devdatta Akhawe ([devd](https://github.com/devd)) |
| Michael Gissing ([scolytus](https://github.com/scolytus)) | Jet Anderson ([thatsjet](https://github.com/thatsjet)) | Dave Wichers ([davewichers](https://github.com/davewichers)) | Jonny Schnittger ([JonnySchnittger](https://github.com/JonnySchnittger)) |
| Silvia Väli ([silviavali](https://github.com/silviavali)) | [jackgates73](https://github.com/jackgates73) | [1songb1rd](https://github.com/1songb1rd) | Timur - ([timurozkul](https://github.com/timurozkul)) |
| Gareth Heyes ([hackvertor](https://github.com/hackvertor)) | [appills](https://github.com/appills) | [suvikaartinen](https://github.com/suvikaartinen) | chaals ([chaals](https://github.com/chaals)) |
| DanielPharos ([AtlasHackert](https://github.com/AtlasHackert)) | will Farrell ([willfarrell](https://github.com/willfarrell)) | Alina Vasiljeva ([avasiljeva](https://github.com/avasiljeva)) | Paul McCann ([ismisepaul](https://github.com/ismisepaul)) |
| Sage ([SajjadPourali](https://github.com/SajjadPourali)) | [rbsec](https://github.com/rbsec) | Benedikt Bauer ([mastacheata](https://github.com/mastacheata)) | James Jardine ([jamesjardine](https://github.com/jamesjardine)) |
| Mark Burnett ([m8urnett](https://github.com/m8urnett)) | [dschwarz91](https://github.com/dschwarz91) | Cyber-AppSec ([Cyber-AppSec](https://github.com/Cyber-AppSec)) | [Tib3rius](https://github.com/Tib3rius) |
| BitnessWise ([bitnesswise](https://github.com/bitnesswise)) | damienbod ([damienbod](https://github.com/damienbod)) | Jared Meit ([jmeit-fwdsec](https://github.com/jmeit-fwdsec)) | Stefan Seelmann ([sseelmann](https://github.com/sseelmann)) |
| Brendan O'Connor ([ussjoin](https://github.com/ussjoin)) | Andrei Titov ([andrettv](https://github.com/andrettv)) | Hans-Petter Fjeld ([atluxity](https://github.com/atluxity)) | [markehack](https://github.com/markehack) |
| Neil Madden ([NeilMadden](https://github.com/NeilMadden)) | Michael Geramb ([mgeramb](https://github.com/mgeramb)) | Osama Elnaggar ([ossie-git](https://github.com/ossie-git)) | [mackowski](https://github.com/mackowski) |
| Ravi Balla ([raviballa](https://github.com/raviballa)) | Hazana ([hazanasec](https://github.com/hazanasec)) | David Means ([dmeans82](https://github.com/dmeans82)) | Alexander Stein ([tohch4](https://github.com/tohch4)) |
| BaeSenseii ([baesenseii](https://github.com/baesenseii)) | Vincent De Schutter ([VincentDS](https://github.com/VincentDS)) | S Bani ([sbani](https://github.com/sbani)) | Mitsuaki Akiyama ([mak1yama](https://github.com/mak1yama)) |
| Christopher Loessl ([hashier](https://github.com/hashier)) | [victorxm](https://github.com/victorxm) | Michal Rada ([michalradacz](https://github.com/michalradacz)) | Veeresh Devireddy ([drveresh](https://github.com/drveresh)) |
| [MaknaSEO](https://github.com/MaknaSEO) | [darkzero2022](https://github.com/darkzero2022) | Liam ([LiamDobbelaere](https://github.com/LiamDobbelaere)) | Frank Denis ([jedisct1](https://github.com/jedisct1)) |
| Otto Sulin ([ottosulin](https://github.com/ottosulin)) | [carllaw6885](https://github.com/carllaw6885) | Anders Johan Holmefjord ([aholmis](https://github.com/aholmis)) | Richard Fritsch ([rfricz](https://github.com/rfricz)) |
| [mesutgungor](https://github.com/mesutgungor) | Scott Helme ([ScottHelme](https://github.com/ScottHelme)) | Carlo Reggiani ([carloreggiani](https://github.com/carloreggiani)) | Suyash Srivastava ([suyash5053](https://github.com/suyash5053)) |
| Mark Potter ([markonweb](https://github.com/markonweb)) | Arjan Lamers ([alamers](https://github.com/alamers)) | Gøran Breivik ([gobrtg](https://github.com/gobrtg)) | [flo-blg](https://github.com/flo-blg) |
| Guillaume Déflache ([guillaume-d](https://github.com/guillaume-d)) | Toufik Airane ([toufik-airane](https://github.com/toufik-airane)) | Keith Hoodlet ([securingdev](https://github.com/securingdev)) | Sinner ([SoftwareSinner](https://github.com/SoftwareSinner)) |
| [iloving](https://github.com/iloving) | Jeroen Beckers ([TheDauntless](https://github.com/TheDauntless)) | Joubin Jabbari ([joubin](https://github.com/joubin)) | yu fujioka ([fujiokayu](https://github.com/fujiokayu)) |
| execjosh ([execjosh](https://github.com/execjosh)) | Alicja Kario ([tomato42](https://github.com/tomato42)) | Sidney Ribeiro ([srjsoftware](https://github.com/srjsoftware)) | Gabriel Marquet ([Gby56](https://github.com/Gby56)) |
| Drew Schulz ([drschulz](https://github.com/drschulz)) | [bedirhan](https://github.com/bedirhan) | [muralito](https://github.com/muralito) | Ronnie Flathers ([ropnop](https://github.com/ropnop)) |
| Philippe De Ryck ([philippederyck](https://github.com/philippederyck)) | Malte ([mal33](https://github.com/mal33)) | [MazeOfThoughts](https://github.com/MazeOfThoughts) | Andreas Falk ([andifalk](https://github.com/andifalk)) |
| Javi ([javixeneize](https://github.com/javixeneize)) | Daniel Hahn ([averell23](https://github.com/averell23)) | [borislav-c](https://github.com/borislav-c) | Robin Wood ([digininja](https://github.com/digininja)) |
| [miro2ns](https://github.com/miro2ns) | Jan Dockx ([jandockx](https://github.com/jandockx)) | [vipinsaini434](https://github.com/vipinsaini434) | [priyanshukumar397](https://github.com/priyanshukumar397) |
| Nat Sakimura ([sakimura](https://github.com/sakimura)) | Benjamin Häublein ([BenjaminHae](https://github.com/BenjaminHae)) | [unknown-user-from](https://github.com/unknown-user-from) | Ali Ramazan TAŞDELEN ([alitasdln](https://github.com/alitasdln)) |
| Pedro Escaleira ([oEscal](https://github.com/oEscal)) | Josh ([josh-hemphill](https://github.com/josh-hemphill)) | Tim Würtele ([SECtim](https://github.com/SECtim)) | AviD ([avidouglen](https://github.com/avidouglen)) |
| SheHacksPurple ([shehackspurple](https://github.com/shehackspurple)) | [fcerullo-cycubix](https://github.com/fcerullo-cycubix) | Hector Eryx Paredes Camacho ([heryxpc](https://github.com/heryxpc)) | Irene Michlin ([irene221b](https://github.com/irene221b)) |
| Jonah Y-M ([TG-Techie](https://github.com/TG-Techie)) | Dhiraj Bahroos ([bahroos](https://github.com/bahroos)) | Jef Meijvis ([jefmeijvis](https://github.com/jefmeijvis)) | [IzmaDoesItbeta](https://github.com/IzmaDoesItbeta) |
| Abdessamad TEMMAR ([TmmmmmR](https://github.com/TmmmmmR)) | [sectroyer](https://github.com/sectroyer) | Soh Satoh ([sohsatoh](https://github.com/sohsatoh)) | [regoravalaz](https://github.com/regoravalaz) |
| james-t ([james-bitherder](https://github.com/james-bitherder)) | Aram Hovsepyan ([aramhovsepyan](https://github.com/aramhovsepyan)) | [JaimeGomezGarciaSan](https://github.com/JaimeGomezGarciaSan) | [ValdiGit01](https://github.com/ValdiGit01) |
| iwatachan ([ishowta](https://github.com/ishowta)) | Vinod Anandan ([VinodAnandan](https://github.com/VinodAnandan)) | Kevin Kien ([KevinKien](https://github.com/KevinKien)) | [paul-williamson-swoop](https://github.com/paul-williamson-swoop) |
| [endergzr](https://github.com/endergzr) | Radhwan Alshamamri ([Rado0z](https://github.com/Rado0z)) | Grant Ongers ([rewtd](https://github.com/rewtd)) | Cure53 ([cure53](https://github.com/cure53)) |
| [AliR2Linux](https://github.com/AliR2Linux) | Ads Dawson ([GangGreenTemperTatum](https://github.com/GangGreenTemperTatum)) | William Reyor ([BillReyor](https://github.com/BillReyor)) | gabe ([gcrow](https://github.com/gcrow)) |
| [mascotter](https://github.com/mascotter) | [luissaiz](https://github.com/luissaiz) | Suren Manukyan ([vx-sec](https://github.com/vx-sec)) | Piotr Gliźniewicz ([pglizniewicz](https://github.com/pglizniewicz)) |
| Tadeusz Wachowski ([tadeuszwachowski](https://github.com/tadeuszwachowski)) | Nasir aka Nate ([andesec](https://github.com/andesec)) | [settantasette](https://github.com/settantasette) | Lars Haulin ([LarsH](https://github.com/LarsH)) |
| Terence Eden ([edent](https://github.com/edent)) | [JasmineScholz](https://github.com/JasmineScholz) | Arun Sivadasan ([teavanist](https://github.com/teavanist)) | Yusuf GÜR ([yusuffgur](https://github.com/yusuffgur)) |
| Troy Marshall ([troymarshall](https://github.com/troymarshall)) | Tanner Prynn ([tprynn](https://github.com/tprynn)) | Nick K. ([nickific](https://github.com/nickific)) | [raoul361](https://github.com/raoul361) |
| Azeem Ilyas ([TheAxZim](https://github.com/TheAxZim)) | Evo Stamatov ([avioli](https://github.com/avioli)) | Tim Potter ([timpotter87](https://github.com/timpotter87)) | Gavin Ray ([GavinRay97](https://github.com/GavinRay97)) |
| monis ([demideus](https://github.com/demideus)) | Marcin Hoppe ([MarcinHoppe](https://github.com/MarcinHoppe)) | Grambulf ([ramshazar](https://github.com/ramshazar)) | Jordan Pike ([computersarebad](https://github.com/computersarebad)) |
| Jason Rogers ([jason-invision](https://github.com/jason-invision)) | Ben Hall ([benbhall](https://github.com/benbhall)) | JamesPoppyCock ([jamesly123](https://github.com/jamesly123)) | WhiteHackLabs ([whitehacklabs](https://github.com/whitehacklabs)) |
| Alex Gaynor ([alex](https://github.com/alex)) | Filip van Laenen ([filipvanlaenen](https://github.com/filipvanlaenen)) | [jeurgen](https://github.com/jeurgen) | [GraoMelo](https://github.com/GraoMelo) |
| Andreas Kurtz ([ay-kay](https://github.com/ay-kay)) | Tom Tervoort ([TomTervoort](https://github.com/TomTervoort)) | old man ([deveras](https://github.com/deveras)) | Marco Schnüriger ([marcortw](https://github.com/marcortw)) |
| [stiiin](https://github.com/stiiin) | infoseclearn ([teaminfoseclearn](https://github.com/teaminfoseclearn)) | [hljupkij](https://github.com/hljupkij) | Noe ([nmarher](https://github.com/nmarher)) |
| Lyz ([lyz-code](https://github.com/lyz-code)) | Martin Riedel ([mrtnrdl](https://github.com/mrtnrdl)) | KIM Jaesuck ([tcaesvk](https://github.com/tcaesvk)) | Barbara Schachner ([bschach](https://github.com/bschach)) |
| René Reuter ([AresSec](https://github.com/AresSec)) | [carhackpils](https://github.com/carhackpils) | Tyler ([tyler2cr](https://github.com/tyler2cr)) | Hugo ([hasousa](https://github.com/hasousa)) |
| Wouter Bloeyaert ([Someniak](https://github.com/Someniak)) | Mark de Rijk ([markderijkinfosec](https://github.com/markderijkinfosec)) | Ramin ([picohub](https://github.com/picohub)) | Philip D. Turner ([philipdturner](https://github.com/philipdturner)) |
| Will Chatham ([willc](https://github.com/willc)) |  |  |  |