Application Security Verification Standard 4.0.3

Final

October 2021

Table of Contents

口絵

本標準について

OWASP 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、利用者がセキュアなアプリケーションの定義、ビルド、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またはテストのリストです。

著作権とライセンス

Version 4.0.3, October 2021

プロジェクトリーダー

Andrew van der Stock	Daniel Cuthbert	Jim Manico
Josh C Grossman	Elar Lang	

主要執筆者

Abhay Bhargav	Benedikt Bauer	Osama Elnaggar
Ralph Andalis	Ron Perris	Sjoerd Langkemper
Tonimir Kisasondi		

その他の執筆者およびレビュー担当者

Aaron Guzman	Alina Vasiljeva	Andreas Kurtz	Anthony Weems	Barbara Schachner
Christian Heinrich	Christopher Loessl	Clément Notin	Dan Cornell	Daniël Geerts
David Clarke	David Johansson	David Quisenberry	Elie Saad	Erlend Oftedal
Fatih Ersinadim	Filip van Laenen	Geoff Baskwill	Glenn ten Cate	Grant Ongers
hello7s	Isaac Lewis	Jacob Salassi	James Sulinski	Jason Axley
Jason Morrow	Javier Dominguez	Jet Anderson	jeurgen	Jim Newman
Jonathan Schnittger	Joseph Kerby	Kelby Ludwig	Lars Haulin	Lewis Ardern
Liam Smit	lyz-code	Marc Aubry	Marco Schnüriger	Mark Burnett
Philippe De Ryck	Ravi Balla	Rick Mitchell	Riotaro Okada	Robin Wood
Rogan Dawes	Ryan Goltry	Sajjad Pourali	Serg Belkommen	Siim Puustusmaa
Ståle Pettersen	Stuart Gunter	Tal Argoni	Tim Hemel	Tomasz Wrobel
Vincent De Schutter	Mike Jang			

クレジットが上記の 4.0.3 クレジットリストにない場合は、将来の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 2008 年の ASVS 1.0 から 2016 年の 3.0 に至る関係者の責任の下で作成されています。今日の ASVS にまだ残っている構成と検証項目の多くはもともと Mike Boberski, Jeff Williams, Dave Wichers によって書かれましたが、もっと多くの執筆者がいます。以前に関わりのあったすべての人々に感謝します。以前のバージョンに貢献したすべての人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 4.0 へようこそ。ASVS は現代の Web アプリケーションおよび Web サービスを設計、開発、テストする際に必要となる、機能的および非機能的なセキュリティ管理策の定義に焦点を当てた、セキュリティ要件および管理策のフレームワークを確立するコミュニティ主導の取り組みです。

バージョン 4.0.3 は v4.0 の 3 番目のマイナーパッチであり、要件の大幅な変更、要件の強化、要件の追加などの重大な変更を加えることなく、スペルミスを修正し、要件を明確にすることを目的としています。ただし、一部の要件は適切と思われる場合には若干弱められ、一部の完全に冗長な要件は削除されています (なお、番号の変更はありません)。

ASVS v4.0 は長年にわたるコミュニティの取り組みと業界のフィードバックの集大成です。セキュアなソフトウェア開発ライフサイクルを通して、さまざまなユースケースに ASVS をより簡単に採用できるようにしました。

ASVS を含むあらゆる Web アプリケーション標準の内容に 100% 合意できるとは考えていません。リスク分析は常にある程度主観的なものであり、さまざまな場面に対応する規格で一般化しようとすることは困難が伴います。しかし、このバージョンで行われた最新の更新が正しい方向への一歩であり、この重要な業界標準に導入されたコンセプトの強化を願っています。

4.0 の更新内容

このバージョンでの最も重要な変更は NIST 800-63-3 デジタルアイデンティティガイドラインの導入であり、最新で、エビデンスをベースとする、高度な認証管理を紹介しています。高度な認証規格との整合については多少の相違が考えられますが、主に他のよく知られているアプリケーションセキュリティ標準規格がエビデンスベースの場合には、標準規格との調整が不可欠と考えています。

準拠する組織が競合する管理策や相反する管理策を決定する必要がないように、情報セキュリティ標準は固有の要件の数を最小限に抑えるように努めるべきです。OWASP Top 10 2017 と現在の OWASP アプリケーションセキュリティ検証標準は認証とセッション管理に関して NIST 800-63 に準拠しています。セキュリティを最大化しコンプライアンスコストを最小化するために、他の標準化団体が私たち、NIST、および他の人たちと協力して、一般に認められている一連のアプリケーションセキュリティ管理策に取り組むことを歓迎します。

ASVS 4.0 は最初から最後まで全面的に番号が付け替えられました。新しい採番スキームにより長期間消えていた章からのギャップを埋めることができ、開発者やチームが順守しなければならない管理策の数を最小限に抑えるために長い章をセグメント化できました。たとえば、アプリケーションが JWT を使用しない場合、セッション管理における JWT のセクション全体が適用されません。

4.0 での新規項目は、近年長期にわたり最もよく求められていた機能要求のひとつ、Common Weakness Enumeration (CWE) への包括的なマッピングです。CWE マッピングにより、ツール開発業者と脆弱性管理ソフトウェアを使用しているユーザは、他のツールおよび以前の ASVS バージョンの結果を 4.0 およびそれ以降に一致させることができます。CWE エントリのためのスペースを確保するために、「導入バージョン」列を廃止しなければなりませんでした。「導入バージョン」列は完全に番号を付け替えたため、以前のバージョンの ASVS のものより意味がありません。ASVS のすべての項目に関連する CWE があるわけではありませんし、CWE には多くの重複があるため、必ずしも近いものではなく最も一般的に使用されているものを使用しようとしています。検証管理は常に同等の脆弱性にマップできるわけではありません。私たちはこのギャップをより一般的なものに埋めることに関して CWE コミュニティおよび情報セキュリティ分野との継続的な議論を歓迎します。

私たちは OWASP Top 10 2017 および OWASP Proactive Controls 2018 に対応する要件を包括的に満たし、上回るように努めてきました。OWASP Top 10 2017 は過失を回避するための最低限のもののため、特定のログ記録を除き意図的にすべての Top 10 要件をレベル 1 管理策とし、OWASP Top 10 の採用者が具体的なセキュリティ標準に容易にステップアップできるようにしています。

ASVS 4.0 レベル 1 は、アプリケーション設計、コーディング、テスト、セキュアコードレビュー、ペネトレーションテストのための PCI DSS 3.2.1 セクション 6.5 の包括的なスーパーセットであることを確認することに着手しました。これには、既存の業界をリードするアプリケーションおよび Web サービスの検証要件に加えて、V5 ではバッファオーバーフローと危険なメモリ操作、V14 では危険なメモリ関連のコンパイラフラグを含める必要がありました。

ASVS はモノリシックなサーバサイドのみの管理策から、現代のすべてのアプリケーションおよび API に対するセキュリティ管理策を提供することへの移行を完了しました。関数型プログラミング、サーバレス API 、モバイル、クラウド、コンテナ、CI/CD および DevSecOps 、フェデレーションなどの時代には、現代のアプリケーションアーキテクチャを無視し続けることはできません。現代のアプリケーションはオリジナルの ASVS が 2009 年にリリースされたときに構築されたものとは全く異なるように設計されています。私たちの主要なオーディエンスである開発者に適切なアドバイスを提供するために、ASVS は常に遠い将来を見据えたものでなければなりません。アプリケーションが単一の組織により所有されているシステム上で実行されることを前提としている要件を明確化または廃止しました。

ASVS 4.0 のサイズ、および他のすべての ASVS の取り組みのためのベースライン ASVS になりたいという私たちの願いのために、OWASP モバイルアプリケーションセキュリティ検証標準(MASVS) に賛同し、モバイルの章を削除しました。Internet of Things の付録は OWASP Internet of Things Project のもとで将来 IoT ASVS として現れるでしょう。付録 C に IoT ASVS の早期プレビューがあります。これらの ASVS をサポートしてくれた OWASP Mobile Team と OWASP IoT Project Team の双方に感謝し、将来彼らが補完的な標準を提供することを楽しみにしています。

最後に、影響の少ない管理策を重複削減および廃止しました。時間とともに、ASVS は包括的な管理策のセットになり始めましたが、すべての管理策がセキュアなソフトウェアを生み出すうえで同等というわけではありません。影響の少ない項目を排除するこの取り組みはさらに進むかもしれません。ASVS の将来のエディションでは、Common Weakness Scoring System (CWSS) が真に重要な管理策と廃止すべき管理策の優先順位付けに役立つでしょう。

バージョン 4.0 では、ASVS は従来および現在のアプリケーションアーキテクチャ、アジャイルセキュリティプラクティス、DevSecOps カルチャーをカバーする、主要な Web アプリおよびサービス標準であることにのみフォーカスしています。

ASVS の使い方

ASVS には主な目標が 2 つあります。

- 組織がセキュアなアプリケーションを開発および保守するのに役立つこと。
- セキュリティサービスベンダ、セキュリティツールベンダ、および利用者が、各々の要件とプロダクトを調整できるようにすること。

アプリケーションセキュリティ検証レベル

アプリケーションセキュリティ検証標準では 3 つのセキュリティ検証レベルを定義しており、レベルごとに深くなっていきます。

- ASVS レベル 1 は低保証レベル向けであり、すべてがペネトレーションテスト可能です。
- ASVS レベル 2 は機密データを含むアプリケーション向けであり、保護を必要とし、ほとんどのアプリに推奨されるレベルです。
- ASVS レベル 3 は極めて重要なアプリケーション向けであり、高額取引を行うアプリケーション、機密性の高い医療データを持つアプリケーション、最高レベルの信頼性を必要とするアプリケーションのためのものです。

各 ASVS レベルはセキュリティ要件のリストを含みます。これらの各要件はセキュリティ固有の機能や開発者がソフトウェアに組み込む必要のある機能にもマップできます。

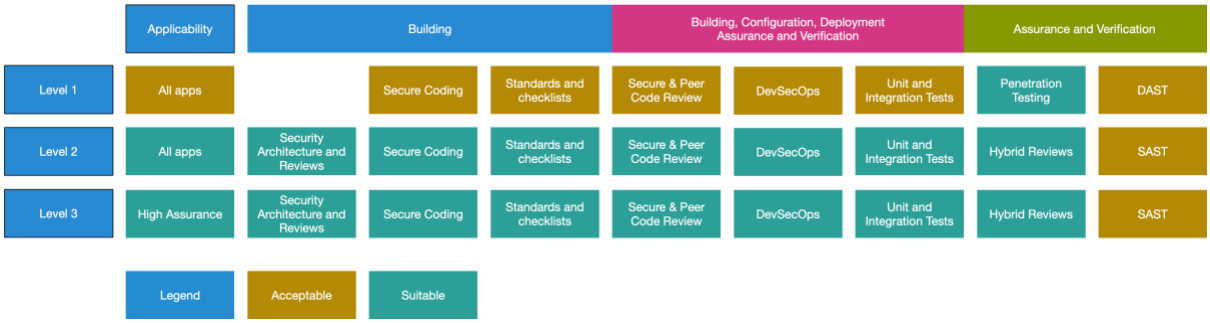

図 1 - OWASP アプリケーションセキュリティ検証標準 4.0 レベル

レベル 1 は人間によりすべてがペネトレーションテスト可能な唯一のレベルです。それ以外のものはすべてドキュメント、ソースコード、設定、開発プロセスに携わる人々へのアクセスを必要とします。但し、たとえレベル 1 が「ブラックボックス」 (ドキュメントなし、ソースなし) テストを行うことができたとしても、それは有効な保証活動ではなく積極的に阻止すべきです。悪意のある攻撃者にはかなりの時間があり、ほとんどのペネトレーションテストは数週間以内に終了します。防御する者はセキュリティ管理策を組み入れ、すべての弱点を保護、発見、解決し、悪意のある行為を行う者を妥当な時間内に検出および対応する必要があります。悪意のある行為を行う者は本質的に無限の時間があり、成功するためにはひとつの侵入しやすい防御、ひとつの弱点、または見逃した検出のみで足ります。ブラックボックステストは開発の最後に行われることが多く、あわただしく行われるか、全く行われず、このような不均衡に完全に対処することはできません。

過去 30 年以上にわたり、ブラックボックステストはさらに深刻な侵害に直接つながる重大なセキュリティ問題を見逃していることが何度も繰り返し証明されてきました。開発プロセス全体を通して開発者とドキュメントへのフルアクセスを伴い、レベル 1 でのペネトレーションテストをソースコード主導 (ハイブリッド) ペネトレーションテストに置き換えるなど、幅広いセキュリティ保証と検証の使用を強く推奨します。金融規制当局は財務記録、サンプル取引、管理を実行する人々にアクセスできない外部の財務監査を認めません。産業界および政府機関はソフトウェア工学の領域で同じ標準の透明性を要求しなければなりません。

開発プロセス自体の中でセキュリティツールを使用することを強く推奨します。DAST および SAST ツールはビルドパイプラインで継続的に使用して、存在してはならないセキュリティ問題を簡単に見つけることができます。

自動ツールとオンラインスキャンは人間による支援なしでは ASVS の半分以上を完了することができません。ビルドごとに包括的なテスト自動化が必要な場合には、カスタムの単体テストと統合テストの組み合わせをビルド開始時のオンラインスキャンとともに使用します。ビジネスロジックの欠陥とアクセス制御のテストは人間による支援でのみ可能です。これらは単体テストと統合テストに変えるべきです。

この標準の使い方

アプリケーションセキュリティ検証標準を使用する最善の方法の 1 つは、アプリケーション、プラットフォーム、組織に固有のセキュアコーディングチェックリストを作成するための青写真として使用することです。ユースケースに合わせて ASVS を仕立て直すことで、プロジェクトや環境にとって最も重要なセキュリティ要件に焦点を当てるでしょう。

レベル 1 - ファーストステップ、自動化、ポートフォリオビュー全体

検出が容易で、OWASP Top 10 や他の同様のチェックリストに含まれているアプリケーションセキュリティ脆弱性に対して適切に防御されていれば、アプリケーションは ASVS レベル 1 を達成します。

レベル 1 はすべてのアプリケーションが目指すべき必要最低限のレベルです。複数フェーズの作業での最初のステップとして、またはアプリケーションが機密データを格納や処理しないためレベル 2 や 3 のより厳しい管理策を必要としない場合にも役立ちます。レベル 1 管理策はツールにより自動的にチェックすることも、ソースコードにアクセスすることなく手動でチェックすることもできます。私たちはレベル 1 をすべてのアプリケーションに最低限必要なものと考えています。

アプリケーションに対する脅威は、発見が容易で悪用が容易な脆弱性を特定するために簡単で手間のかからない技法を使用している攻撃者からのものがほとんどです。これはアプリケーションを明確にターゲットとすることに集中的にエネルギーを費やす、確固たる攻撃者とは対照的です。アプリケーションにより処理されたデータが高い価値を持つ場合には、レベル 1 レビューで止めたくないでしょう。

レベル 2 - 大半のアプリケーション

今日のソフトウェアに関連するリスクのほとんどを適切に防御できれば、アプリケーションは ASVS レベル 2 (または Standard) を達成します。

レベル 2 ではセキュリティ管理策が適用され、効果的であり、アプリケーション内で使用されていることを確認します。ヘルスケア情報の処理、ビジネスに不可欠なまたは機密性の高い機能の実装、他の機密性の高い資産の処理などを含む、重要な企業間取引を処理するアプリケーションや、チートやゲームハックを阻止するゲーム業界などの、ビジネスを保護するために完全性が重要な要素となる業界に対して、通常、レベル 2 が適用されます。

レベル 2 アプリケーションの脅威は、通常、アプリケーション内の弱点を発見および悪用するために高度に実践され効果的なツールや技法を用いて特定のターゲットに集中する、熟練した動機のある攻撃者です。

レベル 3 - 高い価値、高い保証、高い安全性

ASVS レベル 3 は ASVS 内での最高レベルの検証です。このレベルは通常、軍事、安全衛生、重要インフラなどの分野で見られるような、重大なレベルのセキュリティ検証を必要とするアプリケーションを想定しています。

故障が組織の業務に、またその存続可能性にさえ大きく影響する可能性のある、重要な機能を実行するアプリケーションに対して、組織は ASVS レベル 3 を要求する可能性があります。ASVS レベル 3 のアプリケーションに関するガイダンスの例を以下に示します。高度なアプリケーションセキュリティ脆弱性に対して適切に防御し、優れたセキュリティ設計の原則を実証している場合、そのアプリケーションは ASVS レベル 3 (または Advanced) を達成します。

ASVS レベル 3 のアプリケーションは他のすべてのレベルよりもアーキテクチャ、コーディング、テストの詳細な分析を必要とします。セキュアなアプリケーションは (耐性、スケーラビリティ、そして何よりもセキュリティの層を促進するために) 意味のある方法でモジュール化され、各モジュール (ネットワーク接続や物理インスタンスで分離されている) はそれ自身のセキュリティ責任 (多層防御) を管理し、適切に文書化される必要があります。責任には機密性 (暗号化など) 、完全性 (トランザクション、入力検証など) 、可用性 (負荷の適切な処理など) 、認証 (システム間を含む) 、認可、監査 (ログ記録) を確保するための管理策が含まれます。

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。

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.#.# の要件はすべてアーキテクチャの章のものです。
- <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: 1.11.# の要件はすべてアーキテクチャの章のビジネスロジックアーキテクチャ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 4.0.3 での 1.11.3 は以下のとおりです。

認証、セッション管理、アクセス制御などを含む重要度の高いすべてのビジネスロジックフローがスレッドセーフであり、チェック時間と使用時間（TOCTOU）の競合状態に対して耐性がある。

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: v4.0.3-1.11.3 はバージョン 4.0.3 の 'アーキテクチャ' の章の 'ビジネスロジックアーキテクチャ' セクションの 3 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: バージョン部分の前にある v は小文字にします。

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問題が発生することは明らかです。これが記者や開発者がバージョン要素を含めるべき理由です。

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。

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、現在、ベンダ、検証者、ソフトウェアの認証は行っていません。

そのような保証の表明、認証マーク、認証はいずれも OWASP によって公式に検査、登録、認証されたものではありません。そのような見解に依存している組織は ASVS 認証を主張する第三者や認証マークについて、その信頼性に注意する必要があります。

これは、OWASP の公式な認証であると主張しない限り、組織がこのような保証サービスを提供することを妨げるものではありません。

認証機関のためのガイダンス

アプリケーションセキュリティ検証標準はアプリケーションの検証内容が明確化された検証として利用できます。特にレベル 2 とレベル 3 の検証には、設計者や開発者、プロジェクト文書、ソースコード、(役割ごとにひとつ以上のアカウントへのアクセスを含む) テストシステムへの認証されたアクセスといった、主要リソースへのオープンで自由なアクセスを含みます。

歴史上、ペネトレーションテストとソースコードレビューは「例外による」問題を含んでいました。つまり、不合格のテストのみが最終レポートに示されます。認証機関は (特に SSO 認証などの主要コンポーネントがスコープ外の場合) 検証のスコープ、合格および不合格のテストを含む検証結果の要約、不合格のテストへの解決法の明確な指示をレポートに含める必要があります。

特定の検証要件はテスト中のアプリケーションに適用されない場合があります。例えば、クライアント実装なしで顧客にステートレスサービス層 API を提供する場合、V3 セッション管理の要件の多くは直接適用されません。そのような場合、認証機関は依然として ASVS の完全な順守を主張することができますが、そのような除外された検証要件が適用されない理由を明確にレポートに示さなければなりません。

詳細な調書、スクリーンショットやムービー、問題を確実かつ繰り返し利用するためのスクリプト、傍受したプロキシログなどのテストの電子記録、クリーンアップリストなどの関連メモを保持することは標準的な業界の慣行と考えられます。そして、最も疑わしい開発者の発見の証拠として本当に役に立ちます。単にツールを実行して不合格を報告するだけでは十分ではありません。認証レベルのすべての問題がテストされ、余すところなくテストされている十分な証跡を (まったく) 提供していません。異議申し立てがあった場合に備えて、それぞれすべての検証要件が実際にテストされたことを実証するのに十分な証跡が必要となります。

テスト手法

認証機関は適切なテスト手法を自由に選択できますが、レポートに記載する必要があります。

テスト対象のアプリケーションと検証要件に応じて、結果に均一な信頼性を得るためにさまざまなテスト手法を使用できます。例えば、アプリケーションの入力検証メカニズムの有効性を妥当性確認するには、手動ペネトレーションテストで分析するかソースコード解析を用います。

自動セキュリティテストツールの役割

自動ペネトレーションテストツールの使用は、できるだけ多くの範囲をカバーするために推奨されています。

自動ペネトレーションテストツールだけを使用して ASVS 検証を完全に完了することはできません。レベル 1 の要件の大部分は自動テストを使用して実行できますが、要件全体の大半は自動ペネトレーションテストには適していません。

アプリケーションセキュリティ業界が成熟するにつれて、自動テストと手動テストの境界があいまいになっていることに注意してください。自動ツールは専門家により手動で調整されることが多く、手動テスト担当者はさまざまな自動テストツールを活用することがよくあります。

ペネトレーションテストの役割

バージョン 4.0 では、ソースコード、ドキュメント、開発者にアクセスすることなく レベル 1 を完全にペネトレーションテストできるようにしました。OWASP Top 10 2017 A10 に準拠するための必要な二つのログ記録の項目は、OWASP Top 10 2017 の場合と同様に、インタビュー、スクリーンショット、その他の証跡収集が必要になります。しかし、必要な情報にアクセスせずにテストすることは、ソースのレビュー、脅威の特定、管理策の欠如、そしてより短期間での十分なテスト実行の可能性を見逃すため、セキュリティ検証の理想的な方法ではありません。

可能であれば、開発者、ドキュメント、コードへのアクセス、および本番用ではないデータでのテストアプリケーションへのアクセスが、レベル 2 またはレベル 3 アセスメントを実行する際に必要です。これらのレベルで行われるペネトレーションテストには、「ハイブリッドレビュー」や「ハイブリッドペネトレーションテスト」と呼ぶ、このレベルのアクセスが必要です。

ASVS のその他の用途

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される以外に、ASVS 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ひとつはセキュリティアーキテクトのためのリソースです。Sherwood Applied Business Security Architecture (SABSA) にはアプリケーションセキュリティアーキテクチャの十分なレビューを完了するための必要な多くの情報が欠けています。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管理策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。

画一的なセキュアコーディングチェックリストの代わりとして

多くの組織は ASVS を採用することでメリットがあります。3 つのレベルのいずれかを選択するか、ASVS をフォークし、各アプリケーションのリスクレベルに必要なものをドメイン固有の方法で変更します。トレーサビリティが維持されている限りこのタイプのフォークをお勧めします。アプリが要件 4.1 をパスした場合、これはフォークされたコピーについても標準としてそれが進化したものとして同じことを意味します。

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、アーキテクチャ要件および悪性コード要件を除いて、高度にテスト可能なように設計されています。特定の関連するファジングや悪用のケースをテストするユニットテストや統合テストを構築することにより、アプリケーションはそれぞれすべてのビルドごとにほぼ自己検証するようになります。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 ネガティブ項目ではなく、ASVS にある予防的管理策に重点を置いて ASVS を使用できます。

アジャイルアプリケーションセキュリティの牽引役として

ASVS は、セキュアな製品を開発するためにチームが実装する必要がある特定のタスクを定義するためのフレームワークとして、アジャイル開発プロセスで使用できます。ひとつのアプローチとして、レベル 1 から始めて、指定されたレベルの ASVS 要件に従い特定のアプリケーションやシステムを検

証し、どの管理策が欠けているかを見つけ、バックログに特定のチケットやタスクを上げます。これは特定のタスクの優先度付け (または調整) に役立ち、アジャイルプロセスでセキュリティを可視化します。これはまた、特定の ASVS 要件が特定のチームメンバーのレビュー、リファクタリング、監査の牽引役となり、バックログでいずれ行う必要がある「負債」として可視化され、組織内の監査タスクおよびレビュータスクの優先度付けにも使用できます。

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。これは OWASP Secure Software Contract Annex と組み合わせると効果的です。

V1 アーキテクチャ、設計、脅威モデリング

管理目標

セキュリティアーキテクチャは多くの組織で失われた技術となっています。エンタープライズアーキテクトの時代は DevSecOps で過去のものとなりました。アプリケーションセキュリティの分野では、最新のセキュリティアーキテクチャの原則をソフトウェア実務者に再導入しながら、アジャイルセキュリティの原則をキャッチアップし採用する必要があります。アーキテクチャは実装ではなく、潜在的に多くの異なる答えがある可能性があり、唯一の「正しい」答えがない問題について考える方法です。多くの場合、開発者がその問題を解決するはるかに優れた方法を知っている可能性がある場合には、セキュリティは柔軟性がなく、開発者が特定の方法でコードを修正することを要求するものとみなされます。アーキテクチャに対して唯一で単純な解決策はありません。そして、そうではないフリをすることはソフトウェアエンジニアリング分野への害となります。

Web アプリケーションの特定の実装はそのライフタイムを通じて継続的に改訂される可能性がありますが、全体的なアーキテクチャはほとんど変更されず、ゆっくりと進化します。セキュリティアーキテクチャも同様です。私たちは今日認証が必要ですし、明日も認証が必要でしょうし、五年後にも必要でしょう。今日、妥当な判断を下して、アーキテクチャに準拠したソリューションを選択して再利用すれば、多くの労力、時間、費用を節約できます。例えば、一昔前には、多要素認証はほとんど実装されていませんでした。

開発者が SAML フェデレーションアイデンティティなどの単一のセキュアなアイデンティティプロバイダモデルに注力した場合、元のアプリケーションのインタフェースを変更することなく、NIST 800-63 コンプライアンスなどの新しい要件を組み込むためにそのアイデンティティプロバイダを更新できることでしょう。多くのアプリケーションが同じセキュリティアーキテクチャ、つまり同じコンポーネントを共有している場合、すべてのアプリケーションが同時にこのアップグレードの利を得られます。但し、SAML は常に最良ないし最適な認証ソリューションとして残るわけではありません。要件変更に応じて他のソリューションと交換する必要があるかもしれません。このような変更は互いに入り組んでおり、完全な書き直しが必要になるほどコストがかかるか、セキュリティアーキテクチャなしではまったく不可能となります。

本章では、ASVS は妥当なセキュリティアーキテクチャの主要な側面である可用性、機密性、処理の完全性、否認防止、プライバシーをカバーしています。これらの各セキュリティ原則はすべてのアプリケーションに組み込まれ、本質的に備わったものでなければなりません。「シフトレフト」が重要です。セキュアコーディングチェックリスト、メンタリングとトレーニング、コーディングとテスティング、構築、展開、構成、運用で開発者の強化を開始します。そして、すべてのセキュリティ管理策が存在し機能していることを保証するために、フォローアップの独立テストで終了します。かつては業界として私たちが行うすべての作業が最後のステップでしたが、開発者が一日に数十回または数百回コードをプロダクションにプッシュするようになると、それだけでは不十分です。アプリケーションセキュリティの専門家はアジャイル技法に遅れずついていく必要があります。つまり、開発者ツールを採用し、コードを学び、開発者と協力することを意味します。他の全員が異動してから何か月も後にプロジェクトを批判するのではありません。

V1.1 セキュアソフトウェア開発ライフサイクル

#	説明	L1	L2	L3	CWE
1.1.1	開発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用いている。 (C1)		✓	✓	
1.1.2	脅威を特定し、対策を計画し、適切なリスク対応を促進し、セキュリティテストを進めるため、すべての設計変更またはスプリントプランニングに対して脅威モデリングを用いている。		✓	✓	1053

#	説明	L1	L2	L3	CWE
1.1.3	すべてのユーザストーリーおよび機能に、「ユーザが自分のプロファイルを閲覧および編集できるようにする。他のユーザのプロファイルを閲覧または編集できてはいけない」などの機能上のセキュリティ制約が含まれる。		✓	✓	1110
1.1.4	アプリケーションのすべての信頼境界線、コンポーネント、および重要なデータフローのドキュメントと正当性がある。		✓	✓	1059
1.1.5	アプリケーションの高レベルアーキテクチャ、および接続されるすべてのリモートサービスの定義とセキュリティ分析がなされている。 (C1)		✓	✓	1059
1.1.6	重複や欠落がある、非効果的な、もしくはセキュアでない管理策を回避するために、集中管理され、簡潔 (経済的設計) で、徹底調査され、セキュアで、再利用可能なセキュリティ管理策が実装されている。 (C10)		✓	✓	637
1.1.7	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、ガイドラインまたはポリシーを、すべての開発者およびテスト担当者が利用できる。		✓	✓	637

V1.2 認証アーキテクチャ

認証を設計する際、攻撃者がコールセンターを呼び出して一般的に知らせている質問に答えることでアカウントをリセットできる場合、強力なハードウェア対応の多要素認証があるかどうかは問題ではありません。身元を証明するときは、すべての認証経路が同じ強度を持つ必要があります。

#	説明	L1	L2	L3	CWE
1.2.1	すべてのアプリケーションコンポーネント、サービスおよびサーバに対して一意の、または特別な低特権の OS アカウントが使用されている。 (C3)		✓	✓	250
1.2.2	API、ミドルウェア、データ層などのアプリケーションコンポーネント間の通信が認証されている。 コンポーネントには必要最小限の権限が設定されている。 (C3)		✓	✓	306
1.2.3	セキュアであることが知られており、強力な認証を含むよう拡張できる単一で徹底調査された認証機構をアプリケーションが使用していること、アカウントの悪用や侵害を検出するのに十分なログ記録と監視をアプリケーション実装している。		✓	✓	306
1.2.4	アプリケーションのリスクごとに、弱い代替の認証が存在しないように、すべての認証経路と ID 管理 API が一貫した認証セキュリティ制御強度を実装している。		✓	✓	306

V1.3 セッション管理アーキテクチャ

これは将来のアーキテクチャ要件のためのプレースホルダです。

V1.4 アクセス制御アーキテクチャ

#	説明	L1	L2	L3	CWE
1.4.1	アクセス制御ゲートウェイ、サーバ、サーバレス機能などの信頼できる強制ポイントでアクセス制御が実施されている。クライアント上でアクセス制御を実施してはなりません。		✓	✓	602
1.4.2	[削除, 対処不要]				
1.4.3	[削除, 4.1.3 と重複]				
1.4.4	保護されたデータやリソースにアクセスするために、アプリケーションが 1 つの十分に検証されたアクセス制御機構を使用している。コピー&ペーストまたは安全でない代替パスを回避するため、すべてのリクエストがこの単一の機構をパスする必要があります。 (C7)		✓	✓	284
1.4.5	属性または機能ベースのアクセス制御が使用されており、コードは単にユーザのロールではなくむしろ、機能/データ項目に対するユーザの権限を確認します。それでも、権限はロールを利用して割り当てる必要があります。 (C7)		✓	✓	275

V1.5 入力および出力アーキテクチャ

4.0 では、意味のある信頼境界線の用語として「サーバサイド」という用語をなくしました。信頼境界線は依然として重要です。信頼できないブラウザやクライアントデバイスでの決定はバイパス可能です。しかし、今日の主流のアーキテクチャ展開では、信頼の実行点が劇的に変わりました。したがって、ASVS で「信頼できるサービスレイヤ」という用語が使用されている場合、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された API、パートナーAPI や外部 API など、場所に関係なく、信頼された実行点を意味します。

ここでの「信頼できないクライアント」という用語はプレゼンテーション層を提供するクライアントサイドのテクノロジを指し、一般に「フロントエンド」テクノロジと呼ばれます。「シリアライゼーション」という用語は値の配列のようにネットワーク経由でデータを送信することや JSON 構造体を取得して読み取ることだけでなく、ロジックを含む複雑なオブジェクトを渡すことも意味します。

#	説明	L1	L2	L3	CWE
1.5.1	入出力要件が、データの種類や内容および適用法、規制、その他のポリシー準拠に基づいて、取り扱い手順を明確に定義している。		✓	✓	1029
1.5.2	信頼できないクライアントと通信するときにシリアライゼーションが使用されていない。これが不可能な場合は、オブジェクトインジェクションを含むデシリアライゼーション攻撃を防ぐために、適切な完全性制御（および機密データが送信される場合はできる限り暗号化）を実施する。		✓	✓	502
1.5.3	信頼できるサービスレイヤで入力の妥当性確認が実施されている。 (C5)		✓	✓	602
1.5.4	アウトプットエンコーディングが意図されているインタプリタもしくはその近くで生成される。 (C4)		✓	✓	116

V1.6 暗号アーキテクチャ

アプリケーションは分類に従ってデータ資産を保護するために、強力な暗号化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な過失

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や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。暗号を自ら設計したり追加導入したりすることは、最初から単純に組み込むよりも、セキュア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。

アーキテクチャ要件はコードベース全体に内在するため、単体テストや統合テストは困難です。アーキテクチャ要件はコーディングフェーズを通じてコーディング標準を考慮する必要があり、セキュリティアーキテクチャ、ピアレビューやコードレビュー、振り返りの際にレビューする必要があります。

#	説明	L1	L2	L3	CWE
1.6.1	暗号鍵の管理に関する明示的なポリシーがあり、暗号鍵のライフサイクルが NIST SP 800-57 などの鍵管理標準に従っている。		✓	✓	320
1.6.2	暗号化サービスの利用者が、Key Vault または API ベースの代替手段を使用して、キーマテリアルおよびその他の秘密情報を保護している。		✓	✓	320
1.6.3	すべての鍵とパスワードが置き換え可能であり、機密データを再暗号化するため明確に定義されたプロセスの一部となっている。		✓	✓	320
1.6.4	アーキテクチャが対称鍵、パスワード、API トークンなどのクライアントサイドの秘密情報を安全でないものとして扱い、機密データの保護やアクセスにそれらを使用していない。		✓	✓	320

V1.7 エラー、ロギング、監査アーキテクチャ

#	説明	L1	L2	L3	CWE
1.7.1	システム全体で共通のログ形式とアプローチが使用されている。 (C9)		✓	✓	1009
1.7.2	分析、検出、警告、およびエスカレーションのために、できればリモートシステムにログがセキュアに送信されている。 (C9)		✓	✓	

V1.8 データ保護とプライバシーアーキテクチャ

#	説明	L1	L2	L3	CWE
1.8.1	すべての機密データが識別され、保護レベルごとに分類されている。		✓	✓	
1.8.2	暗号化要件、完全性要件、保存期間、プライバシー、その他の機密保持要件などの関連する保護要件一式がすべての保護レベルにあり、それらがアーキテクチャに適用されている。		✓	✓	

V1.9 通信アーキテクチャ

#	説明	L1	L2	L3	CWE
1.9.1	特にコンポーネントが異なるコンテナ、システム、サイトまたはクラウドプロバイダにある場合は、アプリケーションがコンポーネント間の通信を暗号化している。 (C3)		✓	✓	319
1.9.2	中間者攻撃を防ぐために、アプリケーションコンポーネントが通信リンクの両側の信頼性を検証している。例えば、アプリケーションコンポーネントは TLS 証明書とチェーンを検証する必要があります。		✓	✓	295

V1.10 悪意あるソフトウェアのアーキテクチャ

#	説明	L1	L2	L3	CWE
1.10.1	チェックインが、イシューや変更チケットによることを保証するため、手続きを定めてソースコード管理システムが利用されている。ソースコード管理システムは、アクセス制御と識別可能なユーザのみ登録され、どんな変更もトレースできるようにする必要がある。		✓	✓	284

V1.11 ビジネスロジックアーキテクチャ

#	説明	L1	L2	L3	CWE
1.11.1	提供するビジネスまたはセキュリティ機能の観点で、すべてのアプリケーションコンポーネントの定義と文書化がされている。		✓	✓	1059
1.11.2	認証、セッション管理、アクセス制御を含むすべての重要なビジネスロジックフローが非同期状態で共有しない。		✓	✓	362
1.11.3	認証、セッション管理、アクセス制御などを含む重要度の高いすべてのビジネスロジックフローがスレッドセーフであり、チェック時間と使用時間（TOCTOU）の競合状態に対して耐性がある。			✓	367

V1.12 セキュアファイルアップロードアーキテクチャ

#	説明	L1	L2	L3	CWE
1.12.1	[削除, 12.4.1 と重複]				
1.12.2	ユーザがアップロードしたファイル（表示またはアプリケーションからダウンロードする必要がある場合）が、オクテットストリームによるダウンロード、またはクラウドファイルストレージバケットなどの無関係なドメインからダウンロードされる。XSS ベクターまたはアップロードされたファイルによる他の攻撃リスクを軽減するため、適切なコンテンツセキュリティポリシー（Content Security Policy, CSP）が実装されている。		✓	✓	646

V1.13 API アーキテクチャ

これは将来のアーキテクチャ要件のためのプレースホルダです。

V1.14 コンフィギュレーションアーキテクチャ

#	説明	L1	L2	L3	CWE
1.14.1	すべてのアプリケーションコンポーネント、サービス、サーバに対して、一意または特別な低権限のオペレーティングシステムアカウントが使用されている。		✓	✓	923
1.14.2	リモートデバイスにバイナリを展開するために、バイナリ署名、信頼できる接続および検証済みのエンドポイントを使用する。		✓	✓	494
1.14.3	ビルドパイプラインが、古いまたはセキュアでないコンポーネントについて警告し、かつ適切な措置が取られる。		✓	✓	1104

#	説明	L1	L2	L3	CWE
1.14.4	特にクラウド環境のビルドスクリプトなどアプリケーションインフラストラクチャがソフトウェアで定義されている場合、ビルドパイプラインにアプリケーションの安全なデプロイを自動的に構成および検証するビルドステップが含まれている。		✓	✓	
1.14.5	特に機密性が高い、またはデシリアライゼーションなど危険な動作を実行している場合は、攻撃者が他のアプリケーションを攻撃するのを遅らせたり阻止したりするために、アプリケーションのデプロイがネットワークレベルで適切にサンドボックス化、コンテナ化および/または隔離されている。 (C5)		✓	✓	265
1.14.6	NSAPI プラグイン、Flash、Shockwave、ActiveX、Silverlight、NACL、またはクライアントサイドの Java アプレットなど、サポートがされておらずセキュアでない、または非推奨のクライアント側技術をアプリケーションが使用していない。		✓	✓	477

参考情報

詳しくは以下の情報を参照してください。

- OWASP Threat Modeling Cheat Sheet
- OWASP Attack Surface Analysis Cheat Sheet
- OWASP Threat modeling
- OWASP Software Assurance Maturity Model Project
- Microsoft SDL
- NIST SP 800-57

V2 認証

管理目標

認証とは、誰か (あるいは何か) を真正であるとして確立または確認する行為であり、個人またはデバイスの要求は正しく、なりすましに耐性があり、パスワードのリカバリや傍受を防ぐことです。

ASVS が最初にリリースされたとき、ユーザ名 + パスワードは高度なセキュリティシステム以外で最も一般的な認証形式でした。多要素認証 (Multi-factor Authentication, MFA) はセキュリティサークルで一般的に受け入れられていますが、他ではほとんど必要とされていませんでした。パスワード侵害の数が増加するにつれ、ユーザ名は何かしらの形で機密情報であり、パスワードは不明であるという考えでは、多くのセキュリティ管理策は維持できなくなりました。例えば、NIST 800-63 はユーザ名とナレッジベース認証 (Knowledge Based Authentication, KBA) を公開情報、SMS および電子メール通知を「 [制限された](https://pages.nist.gov/800-63-FAQ/#q-b1) 」オーセンティケータタイプ、パスワードをすでに侵害されたものと考えています。この現実は、知識ベースのオーセンティケータ、SMS および電子メールでのリカバリ、パスワード履歴、複雑さ、ローテーション管理が役に立たないと言っています。これらの管理はまったく役に立たず、ユーザは数か月ごとに脆弱なパスワードを考えてきましたが、50 億を超えるユーザ名とパスワード侵害が公表されています。今が前進するときです。

ASVS のすべての章の中で、認証とセッション管理の章が最も変更されています。効果的で、エビデンスベースのリーディングプラクティスの採用は多くの人にとって挑戦となるでしょうが、それはまったくいいことです。今ここでポストパスワードの未来へ、移行を始める必要があります。

NIST 800-63 - 最新のエビデンスベースの認証標準

NIST 800-63b は最新のエビデンスベースの標準であり、適用可能性とは関係なく、利用可能な最適なアドバイスを表しています。この標準は世界中のすべての組織に役立ちますが、特に米国の代理店および米国の代理店を扱う組織に関連します。

NIST 800-63 の用語は、特にユーザ名 + パスワードの認証にしか慣れていない場合には、最初は少しわかりにくいかもしれません。最新の認証には進歩が必要であるため、将来一般的になるであろう用語を導入する必要がありますが、業界がこれらの新しい用語に落ち着くまで理解が難しいことを承知しています。この章の最後に参考のための用語集があります。要件の文字どおりの意味よりも、むしろ要件の目的を満たすために、多くの要件を言い換えました。例えば、NIST が「記憶された秘密 (memorized secret)」を使用する場合、ASVS では「パスワード」という用語を使用します。

ASVS V2 認証、V3 セッション管理、および範囲は狭いですが、V4 アクセス制御は選択された NIST 800-63b 管理策の準拠サブセットに適応し、一般的な脅威と一般的に悪用される認証の脆弱性に焦点を当てています。NIST 800-63 に完全に準拠する必要がある場合には、NIST 800-63 を確認してください。

適切な NIST AAL レベルの選択

アプリケーションセキュリティ検証標準は ASVS レベル 1 を NIST AAL1 要件に、レベル 2 を AAL2 に、レベル 3 を AAL3 にマップしようとしました。しかし、「必須」管理策としての ASVS レベル 1 のアプローチはアプリケーションや API を検証するための正しい AAL レベルとは限りません。例えば、アプリケーションがレベル 3 アプリケーションである場合や AAL3 とする規制要件がある場合には、V2 および V3 セッション管理の章でレベル 3 を選択すべきです。NIST 準拠の認証アサーションレベル (Authentication Assertion Level, AAL) の選択は NIST-63b ガイドラインに従って実行すべきです。 NIST 800-63b Section 6.2 の Selecting AAL に記載されています。

凡例

特に、最新の認証がアプリケーションのロードマップ上にある場合には、アプリケーションは常に現在のレベルの要件を超える可能性があります。以前は、ASVS には必須の MFA を要求していました。NIST は必須の MFA を要求しません。したがって、この章では ASVS が推奨するが管理策を要求しない場所を示すために、オプションの指定を使用しました。この標準では以下の凡例が使用されています。

記号	説明
	必須ではない
o	推奨（必須ではない）
✓	必須

V2.1 パスワードセキュリティ

NIST 800-63 により「記憶された秘密」と呼ばれるパスワードには、パスワード、PIN、ロック解除パターン、正しい子猫や他の画像要素の選択、およびパスフレーズがあります。それらは一般に「あなたが知っているもの（something you know）」と、みなされ多くの場合に一要素オーセンティケータとして使用されます。インターネットで公開されている数十億の有効なユーザ名とパスワード、デフォルトパスワードや脆弱なパスワード、レインボーテーブル、最も一般的なパスワードの順序付き辞書など、一要素認証の継続使用には大きな課題があります。

アプリケーションはユーザに多要素認証への登録を強く推奨し、ユーザが FIDO や U2F トークンなどすでに所有しているトークンを再利用できるようにするか、多要素認証を提供するクレデンシャルサービスプロバイダへのリンクを許可する必要があります。

クレデンシャルサービスプロバイダ (Credential Service Provider, CSP) はユーザにフェデレーション ID (federated identity) を提供します。ユーザは一般的な選択肢として Azure AD, Okta, Ping Identity, Google を使用するエンタープライズ ID や、Facebook, Twitter, Google, WeChat を使用するコンシューマ ID など、複数の CSP で複数の ID を持つことがよくあります。このリストはこれらの企業やサービスを支持するものではなく、多くのユーザが多くの ID をすでに持っているという現実を、開発者が検討することを推奨するものです。組織は CSP の ID プルーフィングの強度のリスクプロファイルに従って、既存のユーザ ID との統合を検討すべきです。例えば、政府機関がソーシャルメディア ID を機密システムのログインとして受け入れることはまずありません。偽の ID を作成、ID を破棄することが簡単なためです。一方、モバイルゲーム会社はアクティブなプレイヤ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と考えるかもしれません。

#	説明	L1	L2	L3	CWE	NIST §
2.1.1	ユーザが設定するパスワードは、(複数の空白が結合された後) 最低 12 文字となっている。(C6)	✓	✓	✓	521	5.1.1.2
2.1.2	64 文字以上のパスワードが許可されている、および 128 文字を超えるパスワードが拒否されている。(C6)	✓	✓	✓	521	5.1.1.2
2.1.3	パスワード切り捨てが行われない。ただし、連続した複数のスペースは単一のスペースに置き換えることができる。(C6)	✓	✓	✓	521	5.1.1.2
2.1.4	スペースや絵文字などの言語ニュートラルな文字を含む、印字可能な Unicode 文字がパスワードに使用できる。	✓	✓	✓	521	5.1.1.2
2.1.5	ユーザは自身のパスワードを変更できる。	✓	✓	✓	620	5.1.1.2

#	説明	L1	L2	L3	CWE	NIST §
2.1.6	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。	✓	✓	✓	620	5.1.1.2
2.1.7	アカウント登録、ログインおよびパスワード変更中に送信されるパスワードが、ローカル（システムのパスワードポリシーに一致する上位 1,000 または 10,000 個の最も一般的なパスワードなど）または外部 API を使用して、侵害されたパスワードと照合される。 API を使用する場合は、平文のパスワードが送信されたりパスワードの侵害状況を確認する際に使用されたりしないように、ゼロ知識証明またはその他の仕組みを使用する必要があります。 パスワードが侵害された場合、アプリケーションはユーザに新しい侵害されていないパスワードの設定を要求する必要があります。(C6)	✓	✓	✓	521	5.1.1.2
2.1.8	ユーザがより強力なパスワードを設定できるように、パスワード強度メータが用意されている。	✓	✓	✓	521	5.1.1.2
2.1.9	使用可能な文字の種類を制限するパスワード規則がない。大文字、小文字、数字、特殊文字を要求する必要はない。(C6)	✓	✓	✓	521	5.1.1.2
2.1.10	定期的なクレデンシャル変更またはパスワード履歴に関する要件がない。	✓	✓	✓	263	5.1.1.2
2.1.11	パスワード入力に対して、ペースト、ブラウザのパスワードヘルパー、および外部パスワードマネージャが使用できる。	✓	✓	✓	521	5.1.1.2
2.1.12	パスワード入力に対して、ペースト、ブラウザのパスワードヘルパー、および外部パスワードマネージャが使用できる。	✓	✓	✓	521	5.1.1.2

注: ユーザにパスワードの表示または最後の一文字の一時的な表示を許可する目的は、特に長いパスワード、パスフレーズ、およびパスワードマネージャの使用に関して、クレデンシャルエントリの使いやすさを向上させることです。この要件を含めるもう一つの理由は、この最新のユーザフレンドリなセキュリティエクスペリエンスを削除するために、組織がビルトインプラットフォームパスワードフィールドの動作をオーバーライドすることを不必要に要求するテストレポートを抑止または防止することです。

V2.2 一般的なオーセンティケータのセキュリティ

オーセンティケータの柔軟性は将来も使い続けるアプリケーションにとって不可欠です。アプリケーションの検証をリファクタリングして、ユーザの好みに応じて追加のオーセンティケータを許可し、非推奨のオーセンティケータや安全でないオーセンティケータを規則正しく廃止できるようにします。

NIST は電子メールや SMS を 「制限された」オーセンティケータタイプ ("restricted" authenticator types) と考えており、将来のある時点で NIST 800-63 および ASVS から削除される可能性があります。アプリケーションは電子メールや SMS の使用を必要としないロードマップを計画すべきです。

#	説明	L1	L2	L3	CWE	NIST §
2.2.1	耐自動化コントロールが流出したクレデンシャルテスト攻撃、ブルートフォース攻撃、およびアカウントロックアウト攻撃の軽減に効果的となっている。このようなコントロールには最も一般的な流出パスワードのブロック、ソフトロックアウト、レート制限、CAPTCHA、試行間の遅延増加、IP アドレスの制限、または場所、デバイスへの最初のログイン、アカウントロック解除の最近の試行などのリスクベースの制限があります。一つのアカウントで一時間あたり 100 回以上の試行失敗が可能ではない。	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2
2.2.2	弱いオーセンティケータ (SMS や電子メールなど) の使用がよりセキュアな認証方式の代わりとしてではなく、二次検証とトランザクション承認に限定されている。より強い方式が弱い方式の前に提示されていること、ユーザがリスクを承知していること、またはアカウント侵害のリスクを制限するために適切な対策が講じられている。	✓	✓	✓	304	5.2.10
2.2.3	クレデンシャルのリセット、電子メールやアドレスの変更、不明な場所や危険な場所からのログインなどの認証詳細の更新後に、セキュアな通知がユーザに送信される。SMS や電子メールではなく、プッシュ通知の使用が推奨されますが、プッシュ通知がない場合、通知に機密情報が開示されていない限り SMS や電子メールは受け入れられます。	✓	✓	✓	620	
2.2.4	多要素認証、目的別暗号化デバイス (プッシュして認証する接続キーなど) 、またはより高い AAL レベルのクライアント側証明書の使用など、フィッシングに対するなりすまし耐性がある。			✓	308	5.2.5
2.2.5	クレデンシャルプロバイダ (Credential Service Provider, CSP) と認証を検証するアプリケーションが分離されている場合、2 つのエンドポイント間で相互に認証された TLS が設定されている。			✓	319	5.2.6
2.2.6	ワンタイムパスワード (One-time Password, OTP) デバイス、暗号化オーセンティケータ、またはルックアップコードを強制的に使用して、リプレイ耐性をつける。			✓	308	5.2.8
2.2.7	OTP トークンの入力や、FIDO ハードウェアキーのボタンを押すなどのユーザ始動のアクションを要求することにより、認証の意思を検証する。			✓	308	5.2.9

V2.3 オーセンティケータライフサイクル

オーセンティケータにはパスワード、ソフトトークン、ハードウェアトークン、生体認証デバイスがあります。オーセンティケータのライフサイクルはアプリケーションのセキュリティにとって重要です。任意の人が身分証明のないアカウントを自己登録できる場合、ID アサーションに対する信頼はほとんどありません。Reddit のようなソーシャルメディアサイトでは、それはまったく問題ありません。銀行システムでは、クレデンシャルやデバイスの登録と発行にフォーカスすることが、アプリケーションのセキュリティにとって重要です。

注: パスワードの最大有効期間をもつべきではありません。パスワードローテーションの原因となります。パスワードは定期的に入れ替えるのではなく、侵害されているかどうかを確認する必要があります。

#	説明	L1	L2	L3	CWE	NIST §
2.3.1	システムが生成した初期パスワードまたはアクティベーションコードは、セキュアでランダムに生成されており、6 文字以上であり、文字と数字を含むことができ、短期間で有効期限が切れる。これらの初期の秘密情報が長期間有効なパスワードになることを許可してはいけない。	✓	✓	✓	330	5.1.1.2 / A.3
2.3.2	U2F や FIDO トークンなど、ユーザが提供する認証デバイスの登録と使用がサポートされている。		✓	✓	308	6.1.3
2.3.3	期限付きのオーセンティケータを更新するとき、十分な時間をとって更新指示が送信される。		✓	✓	287	6.1.4

V2.4 クレデンシャルの保管

アーキテクトおよび開発者はコードをビルドまたはリファクタリングする際にこのセクションを順守すべきです。このセクションはソースコードレビューを使用するか、セキュア単体テストまたはセキュア統合テストを通してのみ、完全に検証できます。ペネトレーションテストではこれらの問題を特定できません。

承認された一方向鍵生成関数のリストは NIST 800-63 B section 5.1.1.2 および BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018) で詳しく説明されています。これらの選択の代わりに、最新の国内または地域のアルゴリズムおよび鍵長標準を選択できます。

このセクションはペネトレーションテストができないため、管理策は レベル 1 に該当しません。ただし、このセクションはクレデンシャルが盗まれた場合に、セキュリティにとって非常に重要であるため、アーキテクチャ、コーディングガイドライン、コードレビューチェックリストに対して ASVS をフォークする場合は、これらの管理策を内部バージョンでは レベル 1 に戻してください。

#	説明	L1	L2	L3	CWE	NIST §
2.4.1	パスワードがオフライン攻撃に強い形式で保存されている。承認済みの一方向鍵生成またはパスワードハッシュ関数を使用して、パスワードをソルト化およびハッシュ化しなければなりません。 鍵生成およびパスワードハッシュ関数は、パスワードハッシュを生成するときに入力として、パスワード、ソルト、およびコストファクターを受け取ります。(C6)		✓	✓	916	5.1.1.2
2.4.2	ソルトの長さが少なくとも 32 ビットであり、保存されているハッシュ間のソルト値の衝突を最小限に抑えるために任意に選択されている。クレデンシャルごとに、固有のソルト値とその結果のハッシュを保管しなければなりません。(C6)		✓	✓	916	5.1.1.2

#	説明	L1	L2	L3	CWE	NIST §
2.4.3	PBKDF2 が使用されている場合、反復回数は検証サーバの性能が許す限り大きくすべきであり、通常少なくとも 100,000 回イテレーションする。(C6)		✓	✓	916	5.1.1.2
2.4.4	bcrypt が使用されている場合、ワークファクターは検証サーバの性能が許す限り大きくする必要があり、10 以上とする。(C6)		✓	✓	916	5.1.1.2
2.4.5	秘密であり検証者のみが知っているソルト値を使用して、鍵生成関数の追加のイテレーションが実行される。承認された乱数ビット生成器 [SP 800-90Ar1] を使用してソルト値を生成し、少なくとも SP 800-131A の最新リビジョンで指定されている最小セキュリティ強度を提供します。秘密のソルト値はハッシュされたパスワードとは別に (例えば、ハードウェアセキュリティモジュールのような専用デバイスに) 保存すべきです。		✓	✓	916	5.1.1.2

米国標準が言及されている場合、必要に応じて米国標準の代わりに、またはそれに加えて、地域またはローカルの標準を使用できます。

V2.5 クレデンシャルリカバリ

#	説明	L1	L2	L3	CWE	NIST §
2.5.1	システムによって生成された初期アクティベーションまたはリカバリ用の秘密情報がユーザに送信されている場合、それはシングルユースで、時間制限があり、ランダムである。(C6)	✓	✓	✓	640	5.1.1.2
2.5.2	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。	✓	✓	✓	640	5.1.1.2
2.5.3	パスワードクレデンシャルのリカバリによって現在のパスワードが明らかにならない。(C6)	✓	✓	✓	640	5.1.1.2
2.5.4	共有アカウントまたはデフォルトアカウントが存在しない（例えば "root", "admin", "sa"）。	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	認証要素が変更または置き換えられた場合、ユーザにこのイベントが通知される。	✓	✓	✓	304	6.1.2.3
2.5.6	パスワードを忘れた場合や他のリカバリパスは、時間ベースの OTP (time-based OTP, TOTP) またはその他のソフトトークン、モバイルパス、または別のオフラインリカバリ機構などのセキュアなリカバリ機構を使用する。(C6)	✓	✓	✓	640	5.1.1.2
2.5.7	OTP または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。		✓	✓	308	6.1.2.3

V2.6 ルックアップシークレット検証者 (Verifier)

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。これらはユーザにセキュアに配布されます。これらのルックアップコードは一度使用され、

すべて使用されると、ルックアップシークレットリストは破棄されます。このタイプのオーセンティケータは「持っているもの (something you have)」とみなされます。

#	説明	L1	L2	L3	CWE	NIST §
2.6.1	ルックアップシークレットは一度だけしか使用されない。		✓	✓	308	5.1.2.2
2.6.2	ルックアップシークレットが十分なランダム性（112 ビットのエントロピー）を持っていること、または、112 ビットのエントロピー以下の場合に、一意でランダムな 32 ビットでソルトされ、承認された一方向ハッシュでハッシュ化されている。		✓	✓	330	5.1.2.2
2.6.3	ルックアップシークレットは予測可能な値などのオフライン攻撃に対して耐性がある。		✓	✓	310	5.1.2.2

V2.7 経路外 (Out of Band) 検証者 (Verifier)

過去において、一般的な経路外オーセンティケータはパスワードリセットリンクを含む電子メールまたは SMS でした。攻撃者はこの脆弱なメカニズムを使用して、ある人物の電子メールアカウントを乗っ取り、発見されたリセットリンクを再利用するなど、まだ制御していないアカウントをリセットします。経路外の検証を処理するためのより良い方法があります。

セキュアな経路外オーセンティケータはセキュアなセカンダリチャネルを介して検証者と通信できる物理デバイスです。例えばモバイルデバイスへのプッシュ通信がそうです。このタイプのオーセンティケータは「持っているもの (something you have)」とみなされます。ユーザが認証したいとき、検証アプリケーションは認証者への接続を介して直接もしくは間接的にサードパーティサービスを通じて経路外オーセンティケータにメッセージを送信します。メッセージには認証コード（通常はランダムな六桁の番号またはモーダル承認ダイアログ）が含まれます。検証アプリケーションはプライマリチャネルを通じて認証コードを受信することを待ち、受信した値のハッシュを元の認証コードのハッシュと比較します。それらが一致する場合、経路外オーセンティケータはユーザが認証されたと想定できます。

ASVS はプッシュ通知などの新たな経路外オーセンティケータを開発する開発者はごく少数であると想定しているため、認証 API 、アプリケーション、シングルサインオン実装などの件奏者に次の ASVS 管理策が適用されます。新しい経路外オーセンティケータを開発する場合には、NIST 800-63B § 5.1.3.1 を参照してください。

電子メールや VOIP などの安全でない経路外オーセンティケータ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、廃止対象であり、プッシュ通知などを推奨しています。電話または SMS の経路外認証を使用する必要がある場合には、§ 5.1.3.3 を参照してください。

#	説明	L1	L2	L3	CWE	NIST §
2.7.1	SMS や PSTN などの平文経路外（NIST で制限されている）オーセンティケータがデフォルトで提供されておらず、プッシュ通知などの強力な代替が最初に提供されている。	✓	✓	✓	287	5.1.3.2
2.7.2	経路外検証者が 10 分後に、帯域外認証リクエスト、コード、またはトークンが期限切れにさせる。	✓	✓	✓	287	5.1.3.2
2.7.3	経路外検証者の認証リクエストやコード、またはトークンが 1 回しか使用できず、元の認証リクエストに対してのみ使用可能となっている。	✓	✓	✓	287	5.1.3.2
2.7.4	経路外オーセンティケータおよび検証者はセキュアで独立したチャネルを介して通信する。	✓	✓	✓	523	5.1.3.2

#	説明	L1	L2	L3	CWE	NIST §
2.7.5	経路外検証者がハッシュ化されたバージョンのオーセンティケーションコードのみを保持している。		✓	✓	256	5.1.3.2
2.7.6	初期オーセンティケーションコードは少なくとも 20 ビットのエントロピーを含む (通常、6 デジタル乱数で十分です) セキュアな乱数生成器により生成されている。		✓	✓	310	5.1.3.2

V2.8 ワンタイム検証者

単一要素ワンタイムパスワード（One-time Password, OTP）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。これらのデバイスはフィッシング (なりすまし) を困難にしますが、不可能ではありません。このタイプのオーセンティケータは「持っているもの (something you have)」とみなされます。多要素トークンは単一要素 OTP 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 OTP を作成する必要があります。

#	説明	L1	L2	L3	CWE	NIST §
2.8.1	時間ベースの OTP は期限切れまでの有効期間が定義されている。	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	送信された OTP を検証するために使用される対称鍵は、ハードウェアセキュリティモジュールまたはセキュアなオペレーティングシステムベースのキーストレージなどを使用して、高度に保護されている。		✓	✓	320	5.1.4.2 / 5.1.5.2
2.8.3	承認済みの暗号化アルゴリズムが OTP の生成、シード、検証に使用されている。		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	時間ベースの OTP が有効期間内に 1 回しか使用できない。		✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	時間ベースの多要素 OTP トークンが有効期間中に再利用された場合、それがログに記録され、セキュアな通知がデバイスの所有者に送信されるとともにリジェクトされる。		✓	✓	287	5.1.5.2
2.8.6	盗難やその他の損失が発生した場合は、物理的な単一要素の OTP ジェネレータを無効にすることができる。場所に関係なく、ログインしたセッション全体で失効がすぐに反映されるようにします。		✓	✓	613	5.2.1
2.8.7	生体認証システムが、自分が持っているものと自分が知っているものの、両方と組み合わせて二次的要素としてのみ使用されるよう制限されている。		o	✓	308	5.2.3

V2.9 暗号化検証者

暗号化セキュリティキーはスマートカードまたは FIDO キーであり、ユーザは認証を完了するために暗号化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。検証者はチャレンジナンスを暗号化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セキュアに保存された暗号化キーに基づいてレスポンスを計算します。

単一要素暗号化デバイスとソフトウェア、および多要素暗号化デバイスとソフトウェアの要件は同じです。これは暗号化オーセンティケータの検証が認証要素の所有を証明するためです。

#	説明	L1	L2	L3	CWE	NIST §
2.9.1	検証に使用される暗号化キーは、Trusted Platform Module (TPM) や Hardware Security Module (HSM) またはこのセキュアなストレージを使用できる OS サービスを使用するなど、セキュアに保存され、漏えいから保護されている。		✓	✓	320	5.1.7.2
2.9.2	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、暗号化デバイスの有効期間を通じて一意となっている。		✓	✓	330	5.1.7.2
2.9.3	承認された暗号化アルゴリズムが生成、シード、および検証に使用されている。		✓	✓	327	5.1.7.2

V2.10 サービス認証

このセクションはペネトレーションテスト可能ではないため、レベル 1 要件はありません。ただし、アーキテクチャ、コーディングまたはセキュアコードレビューで使用する場合、ソフトウェアは (JavaKey Store と同様に) レベル 1 の最小要件と考えてください。どのような状況でも秘密情報の平文での保存は受け入れられません。

#	説明	L1	L2	L3	CWE	NIST §
2.10.1	イントラサービスシークレットが、パスワード、API キーや特権アクセスを備える共有アカウントなどの不変のクレデンシャルに依存していない。		OS assisted	HSM	287	5.1.1.1
2.10.2	サービス認証にパスワードが必要な場合は、使用されるサービスアカウントがデフォルトクレデンシャルではない。 (例えば、あるサービスではインストール時に root/root または admin/admin がデフォルトである)		OS assisted	HSM	255	5.1.1.1
2.10.3	ローカルシステムアクセスを含むオフラインリカバリ攻撃を防ぐために、パスワードは十分保護された状態で保存されている。		OS assisted	HSM	522	5.1.1.1
2.10.4	パスワード、データベースおよびサードパーティシステムとの統合、シードおよび内部シークレット、および API キー がセキュアに管理され、ソースコードに含まれていないこと、またはソースコードリポジトリに保存されていない。このようなストレージはオフライン攻撃に耐える必要があります。パスワードストレージにはセキュアなソフトウェアキーストア (L1) 、ハードウェア TPM 、または HSM (L3) の使用を推奨します。		OS assisted	HSM	798	

追加の米国政府機関要件

米国政府機関には NIST 800-63 に関する必須要件があります。アプリケーションセキュリティ検証標準は常に、アプリのほぼ 100%に適用される管理策の約 80%であり、高度な管理策や適用が制限される管理策の残りの 20%ではありません。そのため、ASVS は特に IAL1/2 および AAL1/2 分類では NIST 800-63 の厳密なサブセットですが、特に IAL3/AAL3 分類に関しては十分に包括的ではありません。

米国政府機関には NIST 800-63 全体をレビューし実装することを強くお勧めします。

用語集

用語	意味
CSP	クレデンシャルサービスプロバイダ (Credential Service Provider) は ID プロバイダ (Identity Provider) とも呼ばれます。
オーセンティケータ (Authenticator)	パスワード、トークン、MFA、フェデレーションアサーションなどを認証するコード。
Verifier	「認証プロトコルを使用して 1 つまたは 2 つのオーセンティケータの認証要求者の所有と制御を検証することにより、認証要求者の身元を検証するエンティティ。これを行うために、検証者はオーセンティケータをサブスクライバ (Subscriber) のオーセンティケータにリンクし、そのステータスを確認するクレデンシャルをバリデートする必要がある場合があります。」
OTP	ワンタイムパスワード (One-time password)
SFA	単一要素オーセンティケータ (Single-factor authenticators) 。知っているもの (記憶された秘密、パスワード、パスフレーズ、PIN) 、持っている特徴 (生体情報、指紋、顔スキャン) 、または持っているもの (OTP トークン、スマートカードなどの暗号化デバイス) など。
MFA	多要素オーセンティケーション (Multi-factor authentication)。2 つ以上の単一要素のものを含む。

参考情報

詳しくは以下の情報を参照してください。

- NIST 800-63 - Digital Identity Guidelines
- NIST 800-63 A - Enrollment and Identity Proofing
- NIST 800-63 B - Authentication and Lifecycle Management
- NIST 800-63 C - Federation and Assertions
- NIST 800-63 FAQ
- OWASP Testing Guide 4.0: Testing for Authentication
- OWASP Cheat Sheet - Password storage
- OWASP Cheat Sheet - Forgot password
- OWASP Cheat Sheet - Choosing and using security questions

V3 セッション管理

管理目標

Web ベースのアプリケーションやステートフル API の中核となるコンポーネントのひとつは、それと対話するユーザやデバイスの状態を制御および保守するメカニズムです。セッション管理はステートレスプロトコルをステートフルに変更します。これはさまざまなユーザやデバイスを区別するために重要です。

検証対象のアプリケーションが以下の上位のセッション管理要件を満たすことを確認します。

- 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ません
- セッションは、不要になったときや非アクティブ期間内にタイムアウトしたときに無効になります

前述のように、これらの要件は、一般的な脅威や一般的に悪用される認証の脆弱性にフォーカスした、選択された NIST 800-63b 管理策の準拠サブセットとなるように適合されています。以前の検証要件は廃止、重複削除、またはほとんどの場合、必須の NIST 800-63b 要件の意図と厳密に一致するように調整されています。

セキュリティ検証要件

V3.1 基本セッション管理セキュリティ

#	説明	L1	L2	L3	CWE	NIST §
3.1.1	アプリケーションが URL パラメータ内にセッショントークンを漏洩しない。	✓	✓	✓	598	

V3.2 セッションバインディング

#	説明	L1	L2	L3	CWE	NIST §
3.2.1	アプリケーションがユーザ認証時に新しいセッショントークンを生成する。 (C6)	✓	✓	✓	384	7.1
3.2.2	セッショントークンに少なくとも 64 ビットのエントロピーがある。 (C6)	✓	✓	✓	331	7.1
3.2.3	適切に保護された Cookie (セクション 3.4 を参照) や HTML 5 セッションストレージなどのセキュアな方法を使用して、アプリケーションがブラウザにセッショントークンのみを保存する。	✓	✓	✓	539	7.1
3.2.4	セッショントークンが承認済みの暗号化アルゴリズムを使用して生成されている。 (C6)		✓	✓	331	7.1

TLS や他のセキュアなトランスポートチャネルはセッション管理の必須要件です。これは通信セキュリティの章でカバーされています。

V3.3 セッションの終了

セッションタイムアウトは NIST 800-63 と整合しています。これはセキュリティ標準で旧来許可されているよりもはるかに長いセッションタイムアウトを許可します。組織は以下の表をレビューする必要があります。また、アプリケーションのリスクに基づいてより長いタイムアウトが望ましい場合には、NIST 値をセッションアイドルタイムアウトの上限にすべきです。

このコンテキストでのレベル 1 は IAL1/AAL1, レベル 2 は IAL2/AAL3, レベル 3 は IAL3/AAL3 です。IAL2/AAL2 および IAL3/AAL3 では、アイドルタイムアウトはより短くなり、ログアウトやセッションを再開するための再認証にはアイドルタイムの下限とします。

#	説明	L1	L2	L3	CWE	NIST §
3.3.1	「戻る」ボタンや下流の依拠当事者 (Relying Parties) が、依拠当事者 (Relying Parties) 間を含め認証済みセッションを再開しないように、ログアウトおよび有効期限によってセッショントークンが無効になる。 (C6)	✓	✓	✓	613	7.1
3.3.2	オーセンティケータがユーザにログインしたままでいることを許可する場合、アクティブに使用されているときと、アイドル期間後の両方で、定期的に再認証が行われる。 (C6)	30 日	12 時間 or 30 分のアイドル状態、2FA はオプショナル	12 時間 or 15 分のアイドル状態、2FA を利用	613	7.2
3.3.3	パスワードが正常に変更 (パスワードリセットやリカバリによる変更を含む) 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。および、これがアプリケーション、フェデレーションログイン（存在する場合）、依拠当事者 (Relying Parties) すべてに有効となっている。		✓	✓	613	
3.3.4	ユーザがすべての現在のアクティブなセッションまたはデバイスを閲覧、および (ログインクレデンシャルを再入力して) ログアウトできる。		✓	✓	613	7.1

V3.4 クッキーベースのセッション管理

#	説明	L1	L2	L3	CWE	NIST §
3.4.1	クッキーベースのセッショントークンに Secure 属性が設定されている。 (C6)	✓	✓	✓	614	7.1.1
3.4.2	クッキーベースのセッショントークンに HttpOnly 属性が設定されている。 (C6)	✓	✓	✓	1004	7.1.1
3.4.3	クロスサイトリクエストフォージェリ攻撃を抑制するために、クッキーベースのセッショントークンが SameSite 属性を利用している。 (C6)	✓	✓	✓	16	7.1.1

#	説明	L1	L2	L3	CWE	NIST §
3.4.4	クッキーベースのセッショントークンに "__Host-" プレフィックスを使用しており、クッキーは最初にクッキーを設定したホストにのみ送信されている。	✓	✓	✓	16	7.1.1
3.4.5	セッションクッキーを開示する可能性があるセッションクッキーを設定または使用する他のアプリケーションを持つドメイン名配下でアプリケーションが公開されている場合は、最も正確なパスを使用してクッキーベースのセッショントークンに path 属性を設定している。 (C6)	✓	✓	✓	16	7.1.1

V3.5 トークンベースのセッション管理

トークンベースのセッション管理には JWT, OAuth, SAML, API キーが含まれます。これらのうち、API キーは脆弱なことが分かっているため、新しいコードでは使用すべきではありません。

#	説明	L1	L2	L3	CWE	NIST §
3.5.1	アプリケーションは、リンクされたアプリケーションとの信頼関係を形成する OAuth トークンを、ユーザが取り消すことができるようにしている。		✓	✓	290	7.1.2
3.5.2	レガシーな実装を除いて、アプリケーションが静的な API シークレットや API キーではなくセッショントークンを使用している。		✓	✓	798	
3.5.3	ステートレスセッショントークンがデジタル署名、暗号化、およびその他の対策を使用して、改ざん、エンベロープピング、リプレイ、ヌル暗号、鍵置き換え攻撃から保護されている。		✓	✓	345	

V3.6 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 Parties, RP）またはクレデンシャルサービスプロバイダ（Credential Service Provider, CSP）のコードを書いている人に関係するものです。これらの機能を実装するコードに依存する場合には、これらの問題が正しく処理されていることを確認します。

#	説明	L1	L2	L3	CWE	NIST §
3.6.1	依拠当事者 (Relying Parties, PR) が Credential Service Provider (CSP) に最大認証時間を指定していること、および CSP がその期間内にセッションを使用していない場合は、CSP がユーザを再認証する。			✓	613	7.2.1
3.6.2	Credential Service Provider (CSP) がユーザを再認証する必要があるかどうかを依拠当事者 (Relying Parties) が決定できるように、CSP が依拠当事者 (Relying Parties) に最後の認証イベントを通知する。			✓	613	7.2.1

V3.7 セッション管理の悪用に対する防御

数は少ないがセッション管理に対する攻撃があり、そのいくつかはセッションのユーザエクスペリエンス（UX）に関連しています。以前は、ISO 27002 要件に基づいて、ASVS は複数の同時セッションをブロックすることを要求していました。最近のユーザは多くのデバイスを使用しているだけでなく、アプリはブラウザセッションを使用しない API であるため、同時セッションのブロックはもはや適切ではありません。これらの実装のほとんどでは、最終認証者が勝利します。それはおおよそ攻撃者です。このセクションでは、コードを使用したセッション管理に対する攻撃の阻止、遅延、検出に関する主要なガイダンスを提供します。

ハーフオープン攻撃の説明

2018 年初頭、攻撃者が「ハーフオープン攻撃」と呼ぶものを使用して、いくつかの金融機関が侵害されました。この用語は業界に受け入れられています。攻撃者はさまざまな独自のコードベースを使用して複数の機関を攻撃しました。実際、同じ機関内でもコードベースが異なるようです。ハーフオープン攻撃は、多くの既存の認証、セッション管理、およびアクセス制御システムで一般的にみられる設計パターンの欠陥を悪用しています。

攻撃者は、クレデンシャルをロック、リセット、リカバリしようと試みることにより、ハーフオープン攻撃を開始します。一般的なセッション管理デザインパターンは、未認証、半認証 (パスワードリセット、ユーザ名忘れ) 、完全認証コードの間でユーザプロファイルセッションオブジェクトやモデルを再利用します。このデザインパターンは、パスワードハッシュやロールなど、被害者のプロファイルを含む有効なセッションオブジェクトやまたはトークンを入力します。コントローラまたはルータのアクセス制御チェックでユーザが完全にログインしていることを正しく検証していない場合、攻撃者はそのユーザとして行動できるかもしれません。攻撃には、ユーザのパスワードを既知の値に変更する、有効なパスワードリセットを実行するために電子メールアドレスを更新する、多要素認証を無効にする、新しい MFA デバイスを登録する、API キーを公開または変更する、などがあります。

#	説明	L1	L2	L3	CWE	NIST §
3.7.1	機密性の高いトランザクションやアカウントの変更を許可する前に、アプリケーションが完全で有効なログインセッションを確保していること、または再認証や二次検証を必要としている。	✓	✓	✓	306	

参考情報

詳しくは以下の情報を参照してください。

- OWASP Testing Guide 4.0: Session Management Testing
- OWASP Session Management Cheat Sheet
- Set-Cookie　Host- prefix details

V4 アクセス制御

管理目標

認可 (Authorization) とは、リソースへのアクセスを、その使用を許可されたユーザのみに制限する概念です。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ことを確認します。

- リソースにアクセスするユーザが有効なクレデンシャルを持つ
- ユーザには、正しく定義された一連のロールと権限が割り当てられている
- Role and permission metadata is protected from replay or tampering.

セキュリティ検証要件

V4.1 一般的なアクセス制御デザイン

#	説明	L1	L2	L3	CWE
4.1.1	特に、クライアント側のアクセス制御が存在し、それが迂回される可能性がある場合には、アプリケーションは信頼できるサービスレイヤに対してアクセス制御ルールを適用する。	✓	✓	✓	602
4.1.2	アクセス制御で使用されるすべてのユーザ属性とデータ属性およびポリシー情報は、特に認可されていない限りエンドユーザーによって操作されない。	✓	✓	✓	639
4.1.3	最小権限の原則が導入されている。ユーザは認可されているものについてのみ、機能やデータファイル、URL、コントローラ、サービス、他のリソースにアクセスできる。これは、なりすましや権限昇格に対する防御となる。(C7)	✓	✓	✓	285
4.1.4	[削除, 4.1.3 と重複]				
4.1.5	例外が発生した場合も含めて、アクセス制御がセキュアに失敗する。(C10)	✓	✓	✓	285

V4.2 オペレーションレベルアクセス制御

#	説明	L1	L2	L3	CWE
4.2.1	機密データおよび API が、他人のレコードの作成や更新、全員のレコードの閲覧、すべてのレコードの削除など、レコードの作成、読み取り、更新、削除を目的とする非セキュアダイレクトオブジェクト参照 (Insecure Direct Object Reference, IDOR) 攻撃に対して保護されている。	✓	✓	✓	639
4.2.2	認証済みの機能を保護するために、アプリケーションやフレームワークが強力な CSRF 対策メカニズムを実施していること、および有効な自動化対策や CSRF 対策が未認証機能の保護を実施している。	✓	✓	✓	352

V4.3 他のアクセス制御の考慮

#	説明	L1	L2	L3	CWE
4.3.1	管理インタフェースが、不正使用を防ぐために、適切な多要素認証を使用している。	✓	✓	✓	419
4.3.2	ディレクトリリスティグは、意図して許可されない限り、無効となっている。また、ファイルやディレクトリのメタデータ (Thumbs.db、.DS_Store、.git、.svn フォルダなど) の検出や開示が許可されていない。	✓	✓	✓	548
4.3.3	アプリケーションが、低価値のシステムを対象とした追加の認可 (ステップアップ認証または適応認証など)や高価値アプリケーションを対象とした職務分掌を備えており、アプリケーションのリスクや過去に行われた不正行為に基づき、不正行為対策の管理策を必須とする。		✓	✓	732

参考情報

詳しくは以下の情報を参照してください。

- OWASP Testing Guide 4.0: Authorization
- OWASP Cheat Sheet: Access Control
- OWASP CSRF Cheat Sheet
- OWASP REST Cheat Sheet

V5 バリデーション、サニタイゼーション、エンコーディング

管理目標

最も一般的な Web アプリケーションのセキュリティ上の弱点は、クライアントや環境からの入力を、出力エンコーディングなしで直接使用する前に適切に検証できないことです。この弱点は、クロスサイトスクリプティング (XSS)、SQL インジェクション、インタプリタインジェクション、ロケール /Unicode 攻撃、ファイルシステム攻撃、バッファオーバーフローなど、Web アプリケーションの重要な脆弱性のほとんどすべてにつながっています。

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ことを確認します。

- 入力のバリデーションと出力エンコーディングのアーキテクチャは、インジェクション攻撃を防ぐために合意されたパイプラインを持っています
- 入力データは強力に型付けされ、バリデーションされ、範囲や長さがチェックされ、最悪の場合、サニタイズされ、フィルタリングされていること
- 出力データは、データのコンテキストに応じて、可能な限りインタプリタに近い形でエンコード、エスケープされること

モダンな Web アプリケーションのアーキテクチャにおいては、出力エンコーディングはこれまで以上に重要です。特定のシナリオでは、堅牢な入力バリデーションを行うことは難しく、パラメータ化されたクエリ、テンプレートフレームワークによる自動エスケープ、または慎重に選択された出力エンコーディングのようなより安全な API を使用することは、アプリケーションのセキュリティにとって非常に重要です。

V5.1 入力バリデーション

ポジティブ許可リストと強力なデータ型付けを使用して適切に実装された入力バリデーション制御は、すべてのインジェクション攻撃の 90%以上を排除することができます。長さと範囲のチェックは、これをさらに減らすことができます。アプリケーションのアーキテクチャ、デザインスプリント、コーディング、ユニットテストと統合テストの間に安全な入力バリデーションの組み込みが要求されます。これらの項目の多くは、ペネトレーションテストでは発見できませんが、実装しなかった場合の結果は、通常、V5.3 (出力エンコーディングとインジェクション防止要件) に記載されています。開発者やセキュアコーディングのレビュー担当者は、インジェクションを防ぐためにすべての項目に L1 が必要であるのと同じように、このセクションを扱うことが推奨されます。

#	説明	L1	L2	L3	CWE
5.1.1	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 (GET、POST、Cookie、ヘッダ、環境など) を区別しない場合はこの防御が必要。	✓	✓	✓	235
5.1.2	フレームワークが大量のパラメータ割り当て攻撃から保護している、またはフィールドを非公開にするなど安全でないパラメータの代入を防ぐための対策が行われている。 (C5)	✓	✓	✓	915
5.1.3	すべての入力データがバリデーションされている。入力データには、HTML のフォームのフィールド、REST 呼び出し、クエリパラメータ、HTTP ヘッダ、Cookie、バッチファイル、RSS フィードなども含む。バリデーションは許可リスト方式で行う。 (C5)	✓	✓	✓	20

#	説明	L1	L2	L3	CWE
5.1.4	構造化データが強く型付けされており、使用可能な文字、長さ、パターン等の定義されたスキーマに基づいてバリデーションされる。（例：クレジットカード番号、電子メールアドレス、電話番号などのバリデーションや、地区名と郵便番号等 2 つの関連するフィールドのデータが妥当なことのバリデーション） (C5)	✓	✓	✓	20
5.1.5	URL のリダイレクト先と転送先がホワイトリストに登録された宛先のみ許可されている、または信頼できない可能性のあるコンテンツにリダイレクトするときに警告を表示する。	✓	✓	✓	601

V5.2 サニタイゼーションとサンドボックス化

#	説明	L1	L2	L3	CWE
5.2.1	WYSIWYG エディタ等から取得した信頼できない HTML 入力が、HTML サニタイザライブラリもしくはフレームワークの機能によって適切にサニタイズされ、入力バリデーションやエンコードにより適切に処理されている。 (C5)	✓	✓	✓	116
5.2.2	非構造化データがサニタイズされ、使用可能な文字や長さなど一般的な対策が適用されている。	✓	✓	✓	138
5.2.3	SMTP または 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する。	✓	✓	✓	147
5.2.4	eval()もしくは動的コード実行機能を使用しない。代替方法がない場合は、実行前に含まれるユーザ入力のサニタイズもしくはサンドボックス化する。	✓	✓	✓	95
5.2.5	テンプレートインジェクション攻撃に対して、ユーザ入力のサニタイズもしくはサンドボックス化によってアプリケーションが保護されている。	✓	✓	✓	94
5.2.6	信頼できないデータやファイル名や URL 入力フィールドなど HTTP ファイルメタデータのバリデーションまたはサニタイズや、プロトコル、ドメイン、パス、ポートに許可リストを使用することで、アプリケーションが SSRF 攻撃から保護されている。	✓	✓	✓	918
5.2.7	ユーザの指定した Scalable Vector Graphics (SVG) スクリプト化可能コンテンツ（特に XSS に関連するインラインスクリプトや foreignObject）に対して、アプリケーションがサニタイズまたはサンドボックス化している。	✓	✓	✓	159
5.2.8	ユーザ指定の Markdown、CSS、XSL スタイルシート、BBCode のようなスクリプト可能もしくは式のテンプレート言語コンテンツに対して、アプリケーションがサニタイズ、無効化またはサンドボックス化されている。	✓	✓	✓	94

V5.3 出力エンコーディングとインジェクション防御

使用しているインタプリタの近くまたは隣接している出力エンコーディングは、アプリケーションのセキュリティにとって非常に重要です。通常、出力エンコーディングは永続化されず、適切な出力コンテキストで出力を安全にレンダリングして、すぐに使用されます。出力エンコードに失敗すると、安全ではなくインジェクション可能で危険なアプリケーションになります。

#	説明	L1	L2	L3	CWE
5.3.1	出力エンコーディングがインタプリタとコンテキストが要求するものに関連している。例えば特に信頼できない入力 ("ねこ" や "O'Hara" など、Unicode 文字やアポストロフィを含む名前など) に対して、HTML 値、HTML 属性、JavaScript、URL パラメータ、HTML ヘッダ、SMTP、その他コンテキストが必要とするものに対して個別のエンコードを使用する。 (C4)	✓	✓	✓	116
5.3.2	どの Unicode 文字でも有効かつ安全に処理されるように、出力エンコーディングがユーザが選択した文字コードセット、ロケールが保持されている。 (C4)	✓	✓	✓	176
5.3.3	コンテキストに応じて (できれば自動化された、あるいは最悪でも手動による) 出力エスケープにより反射型、格納型、および DOM ベース XSS から保護されている。 (C4)	✓	✓	✓	79
5.3.4	データ選択またはデータベースクエリ（例、SQL、HQL、ORM、NoSQL）がクエリのパラメータ化、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、データベースインジェクション攻撃の影響を受けない。 (C3)	✓	✓	✓	89
5.3.5	パラメータ化もしくはより安全な機構が存在しない場合、SQL インジェクションから保護するための SQL エスケープの使用など、コンテキスト固有の出力エンコーディングによりインジェクション攻撃から保護されている。 (C3, C4)	✓	✓	✓	89
5.3.6	アプリケーションが JSON インジェクション攻撃、JSON eval 攻撃、および JavaScript 式評価から保護されている。 (C4)	✓	✓	✓	830
5.3.7	アプリケーションが LDAP インジェクションの影響を受けない、またはセキュリティ管理策によって LDAP インジェクションが防止される。 (C4)	✓	✓	✓	90
5.3.8	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、もしくはコンテキストコマンドライン出力エンコーディングを使用する。 (C4)	✓	✓	✓	78
5.3.9	アプリケーションが、リモートファイルインクルード (RFI) やローカルファイルインクルード (LFI) の影響を受けない。	✓	✓	✓	829
5.3.10	アプリケーションが XPath インジェクション攻撃や XML インジェクション攻撃から保護されている。 (C4)	✓	✓	✓	643

注:クエリのパラメータ化や 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 など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作成データが含まれていると、クエリの失敗や SQL インジェクションが発生します。

注: SVG フォーマットは、ほとんどすべてのコンテキストで ECMA スクリプトを明示的に許可しているため、すべての SVG XSS ベクターを完全にブロックすることは不可能かもしれません。SVG のアップロードが必要な場合は、アップロードされたファイルを text/plane として提供するか、別のユーザ指定コンテンツドメインを使用して、アプリケーションから引き継がれた XSS を防止します。

V5.4 メモリ、文字列、アンマネージドコード

以下の要件は、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している場合にのみ適用されます。

#	説明	L1	L2	L3	CWE
5.4.1	メモリセーフな文字列、安全なメモリコピー、ポインタ演算を使って、スタック、バッファ、ヒープのオーバーフローをアプリケーションが検出または防止する。		✓	✓	120
5.4.2	フォーマット文字列は悪意のある入力を受け取らず、定数となっている。		✓	✓	134
5.4.3	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が使用されている。		✓	✓	190

V5.5 デシリアライゼーション防御

#	説明	L1	L2	L3	CWE
5.5.1	シリアライズされたオブジェクトが整合性チェックを使用していること、または悪意のあるオブジェクトの作成やデータの改ざんを防ぐために暗号化されている。 (C5)	✓	✓	✓	502
5.5.2	アプリケーションが XML パーサを可能な限り最も制限の厳しい構成のみを使用するように正しく制限し、外部エンティティの解決などの危険な機能を無効にして XML 外部エンティティ (XML eXternal Entity, XXE) を防ぐようにしている。	✓	✓	✓	611
5.5.3	信頼できないデータのデシリアライズが回避されている、またはカスタムコードとサードパーティのライブラリ（JSON、XML、YAML パーサなど）の両方で保護されている。	✓	✓	✓	502
5.5.4	ブラウザもしくは JavaScript ベースのバックエンドで JSON をパースするときは JSON.parse を使用する。eval() を使用しない。	✓	✓	✓	95

参考情報

詳しくは以下の情報を参照してください。

- OWASP Testing Guide 4.0: Input Validation Testing
- OWASP Cheat Sheet: Input Validation
- OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution
- OWASP LDAP Injection Cheat Sheet
- OWASP Testing Guide 4.0: Client Side Testing
- OWASP Cross Site Scripting Prevention Cheat Sheet
- OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet
- OWASP Java Encoding Project
- OWASP Mass Assignment Prevention Cheat Sheet
- DOMPurify - Client-side HTML Sanitization Library
- XML External Entity (XXE) Prevention Cheat Sheet

自動エスケープの詳細な情報はこちらをご覧ください。

- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems
- AngularJS Strict Contextual Escaping
- AngularJS ngBind
- Angular Sanitization
- Angular Security
- ReactJS Escaping
- Improperly Controlled Modification of Dynamically-Determined Object Attributes

デシリアライゼーションの詳細情報はこちらをご覧ください。

- OWASP Deserialization Cheat Sheet
- OWASP Deserialization of Untrusted Data Guide

V6 保存時の暗号化

管理目標

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ことを確認します。

- すべての暗号モジュールが安全な方法で失敗し、エラーが正しく処理されること
- 適切な乱数発生器が使用されていること
- 鍵へのアクセスが安全に管理されていること

V6.1 データ分類

最も重要な資産は、アプリケーションによって処理、保存、または送信されたデータです。保存されているデータのデータ保護の必要性を正しく分類するために、常にプライバシー影響評価を実施してください。

#	説明	L1	L2	L3	CWE
6.1.1	個人を特定できる情報（Personally Identifiable Information, PII）、センシティブな個人情報、または EU の GDPR の対象となる可能性が高いと判断されたデータなど、規制対象の非公開データが暗号化されて保管されている。		✓	✓	311
6.1.2	医療記録、医療機器の詳細、匿名化されていない調査記録など、規制対象の医療データが暗号化されて保存されている。		✓	✓	311
6.1.3	金融口座、債務不履行やクレジットの履歴、税務記録、支払い履歴、受益者、匿名化されていない市場や調査記録など、規制対象の財務データが暗号化されて保管されている。		✓	✓	311

V6.2 アルゴリズム

最近の暗号技術の進歩は、以前は安全だったアルゴリズムや鍵の長さが、データを保護するためにはもはや安全でも十分でもないことを意味しています。したがって、アルゴリズムを変更することは必要です。

このセクションはペネトレーションテストが難しく、ほとんどの項目で L1 が含まれていませんが、開発者はこのセクション全体を必須項目と考えてください。

#	説明	L1	L2	L3	CWE
6.2.1	すべての暗号化モジュールにおいて、処理に失敗した場合の安全対策が施されており、オラクルパディング攻撃を許さない方法でエラーが処理される。	✓	✓	✓	310
6.2.2	独自実装された暗号化方式ではなく、業界で実績のある、または政府が承認した暗号化アルゴリズム、モード、およびライブラリが使用されている。 (C8)		✓	✓	327
6.2.3	最新の勧告を使用して、暗号初期化ベクトル、暗号設定およびブロックモードが安全に構成されている。		✓	✓	326
6.2.4	暗号の破壊から保護するため、乱数、暗号化またはハッシュアルゴリズム、鍵の長さ、ラウンド、暗号方式またはモードが、いつでも再構成やアップグレードもしくは入れ替えられる。 (C8)		✓	✓	326

#	説明	L1	L2	L3	CWE
6.2.5	既知の安全でないブロックモード（ECB など）、パディングモード（PKCS#1 v1.5 など）、ブロックサイズが小さい暗号（Triple-DES、Blowfish など）、および弱いハッシュアルゴリズム（MD5、SHA1 など）が、下位互換のために要求されない限り使用されていない。		✓	✓	326
6.2.6	ナンス、初期化ベクトル、およびその他の使い捨ての数字が、特定の暗号化鍵で複数回使いまわしされていない。生成方法は、使用しているアルゴリズムに適切となっている。		✓	✓	326
6.2.7	暗号化されたデータが署名、認証された暗号モード、または HMAC によって認証され、暗号文が権限のない者によって変更されない。			✓	326
6.2.8	情報の漏洩を防ぐために、すべての暗号化操作が、比較や計算または返戻の際にショートカット操作がなく、一定時間となっている。			✓	385

V6.3 乱数値

真の疑似乱数生成 (Pseudo-random Number Generation, PRNG) 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良い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予測可能な鍵や秘密情報につながる可能性があります。

#	説明	L1	L2	L3	CWE
6.3.1	すべての乱数、ランダムなファイル名、ランダムな GUID、ランダムな文字列は、攻撃者が推測できないことを意図している場合、暗号化モジュールが許可する乱数生成器を用いて生成する。		✓	✓	338
6.3.2	ランダムな GUID が GUID v4 アルゴリズムと暗号的に安全な疑似乱数ジェネレータ（Cryptographically-secure Pseudo-random Number Generator, CSPRNG）を使用して作成されている。他の疑似乱数ジェネレータを使用して作成された GUID は予測可能な場合がある。		✓	✓	338
6.3.3	アプリケーションの負荷が高い状態であっても、乱数が適切なエントロピーを使って生成される、あるいはアプリケーションがそのような環境で適切にデグレードする。			✓	338

V6.4 シークレット管理

このセクションはペネトレーションテストが難しく、ほとんどの項目で L1 が含まれていませんが、開発者はこのセクション全体を必須項目と考えてください。

#	説明	L1	L2	L3	CWE
6.4.1	鍵保管庫のような秘密情報管理ソリューションを、秘密情報の作成、保存、アクセス制御および秘密情報の破壊に使用している。 (C8)		✓	✓	798
6.4.2	鍵マテリアルがアプリケーションに公開されていない。代わりに暗号化操作用の金庫のように隔離されたセキュリティモジュールを使用している。 (C8)		✓	✓	320

参考情報

詳しくは以下の情報を参照してください。

- OWASP Testing Guide 4.0: Testing for weak Cryptography
- OWASP Cheat Sheet: Cryptographic Storage
- FIPS 140-2

V7 エラー処理とログ記録

管理目標

エラー処理とログ記録の主な目的は、ユーザ、管理者、インシデント対応チームに有用な情報を提供することです。目的は、大量のログを作成することではなく、廃棄されるノイズよりも多くのシグナルを含む高品質のログを作成することです。

高品質のログには機密データが含まれていることが多く、現地のデータプライバシー法や指令に従って保護されなければなりません。これには以下が含まれます。

- 特に必要とされない限り、機密情報の収集やロギングを行わないこと
- すべてのログ情報が安全に処理され、データ分類に従って保護されていること
- ログは永久に保存せず、可能な限り短い絶対的な寿命を持つようにすること

定義は国によって異なりますが、ログに個人情報や機密性の高いデータが含まれている場合、ログはアプリケーショ ンによって保持される最も機密性の高い情報の一部となり、攻撃者にとって非常に魅力的なものとなります。

また、アプリケーションが安全に失敗し、エラーが不必要な情報を開示しないようにすることも重要です。

V7.1 ログ内容

機密情報をログに記録するのは危険です。ログそのものが機密情報に分類されるため、暗号化する必要が生じ、保存ポリシーの対象となり、セキュリティ監査で開示する必要があります。必要な情報のみをログに保存し、支払い、クレデンシャル（セッショントークンを含む）、機密情報または個人を特定できる情報は絶対に含めないでください。

V7.1 は OWASP Top 10 2017:A10 をカバーしています。2017:A10 とこのセクションはペネトレーションテストが不可能なため、以下の点が重要です。

- 開発者は、このセクションで L1 とマークされているすべての項目に完全準拠すること
- ペネトレーションテスト担当者は、インタビューやスクリーンショットまたはアサーションを介して V7.1 のすべての項目への完全準拠を検証すること

#	説明	L1	L2	L3	CWE
7.1.1	アプリケーションがクレデンシャルまたは支払い情報をログに保存していない。セッショントークンは、不可逆的なハッシュ形式でのみログに保存する。 (C9, C10)	✓	✓	✓	532
7.1.2	現地のプライバシー法または関連するセキュリティポリシーで定義されているその他のセンシティブなデータをログに記録していない。 (C9)	✓	✓	✓	532
7.1.3	成功および失敗した認証イベント、アクセス制御の失敗、デシリアライゼーションの失敗、および入力検証の失敗を含むセキュリティへの影響が考えられるイベントについてログを保存する。 (C5, C7)		✓	✓	778
7.1.4	各ログのイベントには、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情報が含まれている。 (C9)		✓	✓	778

V7.2 ログ処理

タイムリーなログ記録は、監査イベント、トリアージおよびエスカレーションにとって重要です。アプリケーションのログが明確で、ローカルまたはリモートの監視システムに送信されたログのいずれかで簡単に監視および分析できることを確認してください。

V7.2 は OWASP Top 10 2017:A10 をカバーしています。2017:A10 とこのセクションはペネトレーションテストが不可能なため、以下の点が重要です。

- 開発者は、このセクションで L1 とマークされているすべての項目に完全準拠すること
- ペネトレーションテスト担当者は、インタビューやスクリーンショットまたはアサーションを介して V7.2 のすべての項目への完全準拠を検証すること

#	説明	L1	L2	L3	CWE
7.2.1	センシティブなセッショントークンやパスワードを保存せずに、すべての認証判定がログに記録されている。これには、セキュリティ調査に必要な関連メタデータを含むリクエストを含める。		✓	✓	778
7.2.2	すべてのアクセス制御判定がログに記録され、失敗したすべての判定がログに記録される。これには、セキュリティ調査に必要な関連メタデータを含むリクエストを含める。		✓	✓	285

V7.3 ログ保護

簡単に変更や削除が可能なログは、調査や訴追には使えません。ログの公開により、アプリケーションまたはアプリケーションに含まれるデータに関する内部の詳細が明らかになる可能性があります。不正な開示、変更、削除からログを保護する際には注意が必要です。

#	説明	L1	L2	L3	CWE
7.3.1	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。 (C9)		✓	✓	117
7.3.2	[削除, 7.3.1 と重複]				
7.3.3	セキュリティログが不正なアクセスや改変から保護されている。 (C9)		✓	✓	200
7.3.4	時刻源が正しい時間と正しいタイムゾーンに同期されている。システムがグローバルである場合は、インシデント後のフォレンジック分析を支援するために UTC でのみログを記録することを強く検討する。 (C9)		✓	✓	

注：ログのエンコーディング（7.3.1）は、自動化された動的ツールやペネトレーションテストによるテストやレビューは困難ですが、設計者、開発者、ソースコードレビュー担当者はログのエンコーディングを L1 の要件と考える必要があります。

V7.4 エラー処理

エラー処理の目的は、アプリケーションが監視やトリアージおよびエスカレーションのためにセキュリティ関連のイベントを提供できるようにすることです。目的はログを作成することではありません。セキュリティ関連のイベントをログに記録するときは、ログに目的があること、および SIEM または分析ソフトウェアによって区別できることを確認してください。

#	説明	L1	L2	L3	CWE
7.4.1	予期しないエラーまたはセキュリティ上重要なエラーが発生したときに、サポート担当者が調査に使用できる一意の ID とともに一般的なメッセージが表示される。 (C10)	✓	✓	✓	210
7.4.2	予期されるエラーおよび予期されないエラー状態を説明するために、例外処理（または機能的に同等なもの）がコードベース全体で使用されている。 (C10)		✓	✓	544
7.4.3	未処理の例外をすべて捕捉する「最後の手段」となるエラーハンドラが定義されている。 (C10)		✓	✓	431

注：Swift や Go のような特定の言語や（および一般的な設計手法を通して）多くの関数型言語は、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、設計者と開発者は、パターン、言語またはフレームワークで簡単に使える方法を使用して、例外や予期しないイベントまたは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 Testing Guide 4.0 content: Testing for Error Handling
- OWASP Authentication Cheat Sheet section about error messages

V8 データ保護

管理目標

データ保護を適切に行うには次の 3 つの要素を考慮する必要があります。機密性（Confidentiality）、完全性（Integrity）、可用性（Availability）の CIA です。この標準ではデータ保護が、強固かつ十分な保護を備えるサーバなど信頼できるシステム上でデータ保護が行われていることを想定しています。

アプリケーションは「ユーザデバイスはどれも完全には信頼できない」ことを前提にする必要があります。共有コンピュータや電話、タブレット等の安全でないデバイスに対してセンシティブなデータの送信や保存を行う場合、アプリケーション側が責任を持って、デバイス上に保存するデータを暗号化し、不正な取得や変更、開示が容易にはできないよう保護する必要があります。

検証対象のアプリケーションが以下の上位のデータ保護要件を満たすことを確認します。

- 機密性：送信と保存の両方で認可されていない監視や開示からデータが保護されている
- 完全性：攻撃者による悪意のある作成，変更，削除からデータが保護されている
- 可用性：必要なときにデータが許可されたユーザに提供される

V8.1 一般的なデータ保護

#	説明	L1	L2	L3	CWE
8.1.1	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保護している。		✓	✓	524
8.1.2	サーバ上に保存されている機密データの、すべてのキャッシュや一時コピーが、認証されていないアクセスから保護されているか、認証されたユーザが機密データにアクセスした後に削除または無効化される。		✓	✓	524
8.1.3	1 リクエスト中に含まれるパラメータの数（非表示フィールド、Ajax 変数、Cookie、ヘッダ値など）を最小限にしている。		✓	✓	233
8.1.4	アプリケーションが、IP 、ユーザ、1 時間または 1 日あたりの総数、またはアプリケーションにとって重要なリクエストなど、異常な数のリクエストを検出および警告できる。		✓	✓	770
8.1.5	重要なデータの定期的なバックアップが実行され、データのリストアテストが実行されている。			✓	19
8.1.6	データの盗難や破損を防ぐために、バックアップがセキュアに保存されている。			✓	19

V8.2 クライアントサイドのデータ保護

#	説明	L1	L2	L3	CWE
8.2.1	最新のブラウザで機密データが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ヘッダを設定する。	✓	✓	✓	525
8.2.2	ブラウザの記憶域（localStorage、sessionStorage、IndexedDB、Cookie など）に保存されるデータにセンシティブなデータが含まれていない。	✓	✓	✓	922

#	説明	L1	L2	L3	CWE
8.2.3	セッションの終了後、ブラウザの DOM など認証されたデータがクライアントの記憶域から消去される。	✓	✓	✓	922

V8.3 機密性の高い個人データ

このセクションは、特に大量の場合、センシティブなデータを認証なく作成、読み取り、更新、削除することから保護するのに役立ちます。

このセクションへの準拠は V4 アクセス制御、特に V4.2 への準拠を意味します。例えば、認証のない更新やセンシティブな個人情報の開示から保護するには V4.2.1 を順守する必要があります。完全に網羅するにはこのセクションと V4 に従ってください。

注: オーストラリアのプライバシー原則 APP-11 や GDPR などのプライバシー規制や法令は、センシティブな個人情報の保存、使用、および転送の実装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直接影響します。これには厳しいペナルティから簡単なアドバイスまであります。地域の法令や規制を調べ、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	L1	L2	L3	CWE
8.3.1	センシティブなデータが HTTP メッセージボディまたはヘッダでサーバに送信される。どんな HTTP verb のクエリストリングパラメータにもセンシティブなデータが含まれない。	✓	✓	✓	319
8.3.2	ユーザが自身のデータをオンデマンドで、削除またはエクスポートできる。	✓	✓	✓	212
8.3.3	個人情報の収集および利用に関する明確なポリシーが、ユーザに提供されている。個人情報が何らかの方法で使用される前に、オプトイン方式で個人情報利用に関する同意を得ている。	✓	✓	✓	285
8.3.4	アプリケーションにより作成および処理される、すべてのセンシティブなデータを特定している。センシティブなデータを処理する方法に関するポリシーが、策定されている。 (C8)	✓	✓	✓	200
8.3.5	データが関連するデータ保護規制の下で収集されている場合や、アクセスログ記録が必要な場合、センシティブなデータへのアクセスが（センシティブなデータ自体がログに記録されることなく）監査されている。		✓	✓	532
8.3.6	メモリダンプ攻撃を軽減するため、メモリに保持されたセンシティブな情報は、不要になればすぐに、ゼロまたはランダムデータを使用して上書きする。		✓	✓	226
8.3.7	暗号化を必要とするセンシティブなもしくは個人情報は、機密性と完全性の双方を提供する承認済みアルゴリズムを使用して暗号化する。 (C8)		✓	✓	327
8.3.8	センシティブな個人情報は、古いデータや期限切れのデータが、自動的、定期的、または状況に応じて削除されるなど、データ保持分類の対象となっている。		✓	✓	285

データ保護を検討する際には、主に一括抽出、一括変更、過度の使用について考慮すべきです。例えば、多くのソーシャルメディアシステムではユーザは 1 日に新しい友人を 100 人しか追加できませんが、これらのリクエストがどのシステムから来たのかは重要ではありません。銀行のプラットフォー

ムでは、1000 ユーロを超える資金を外部の機関に転送することは、1 時間当たり 5 つの取引まででブロックすることが期待されています。各システムの要件は大きく異なることがあるため、「異常」を判断する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。重要な基準はそのような異常な大量のアクションを検出、阻止、または可能であればブロックする能力です。

参考情報

詳しくは以下の情報を参照してください。

- Consider using Security Headers website to check security and anti-caching headers
- OWASP Secure Headers project
- OWASP Privacy Risks Project
- OWASP User Privacy Protection Cheat Sheet
- European Union General Data Protection Regulation (GDPR) overview
- European Union Data Protection Supervisor - Internet Privacy Engineering Network

V9 通信

管理目標

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ことを確認します。

- コンテンツの機密性に関わらず、TLS または強力な暗号を必要としている。
- 以下のような最新のガイダンスに従っている。
  - 構成に関する勧告
  - 優先アルゴリズムおよび暗号
- 最後の手段を除いて、脆弱または近い将来廃止予定のアルゴリズムや暗号を避けている。
- 非推奨または既知のセキュアでないアルゴリズムや暗号を無効にしている。

これらの要件を満たすために:

- セキュアな TLS 構成に関する業界の推奨事項は、(多くの場合、既存のアルゴリズムや暗号の壊滅的な破綻が原因で) 頻繁に変更されるため、最新の状態に保ちます。
- 最新バージョンの TLS 設定レビューツールを使用して、優先順位とアルゴリズム選択を設定します。
- 設定を定期的にチェックして、セキュアな通信が常に存在し有効であることを確認します。

V9.1 クライアント通信のセキュリティ

すべてのクライアントメッセージは TLS 1.2 以降を使用して、暗号化されたネットワーク上で送信されるようにします。最新のツールを使用して、クライアント構成を定期的に見直します。

#	説明	L1	L2	L3	CWE
9.1.1	TLS がすべてのクライアント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。 (C8)	✓	✓	✓	319
9.1.2	最新の TLS テストツールを使用して、強力な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。	✓	✓	✓	326
9.1.3	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。	✓	✓	✓	326

V9.2 サーバ通信のセキュリティ

サーバ通信は HTTP だけではありません。監視システム、管理ツール、リモートアクセスや SSH、ミドルウェア、データベース、メインフレーム、パートナーまたは外部ソースシステムなど、他のシステムとの安全な接続が必要です。「外部は堅牢だが、内部が傍受しやすい」ことを防ぐために、これらの通信はすべて暗号化する必要があります。

#	説明	L1	L2	L3	CWE
9.2.1	サーバ接続（外向き、内向きの両方）において、信頼できる TLS 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、サーバ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し、それ以外の証明書はすべて拒否する。		✓	✓	295
9.2.2	管理ポート、監視、認証、API、または Web サービスの呼び出し、データベース、クラウド、サーバレス、メインフレーム、外部やパートナー間の接続など、すべての外向き／内向きの接続に TLS などの暗号化通信が使用されている。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。		✓	✓	319
9.2.3	センシティブな情報や機能を持つ外部システムとの暗号化接続がすべて認証されている。		✓	✓	287
9.2.4	Online Certificate Status Protocol (OCSP) Stapling など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。		✓	✓	299
9.2.5	バックエンドの TLS 通信に関するエラーがログに記録されている。			✓	544

参考情報

詳しくは以下の情報を参照してください。

- OWASP – TLS Cheat Sheet
- OWASP - Pinning Guide
- 「TLS の承認されたモード」に関する注記:
  - これまで ASVS は米国標準 FIPS 140-2 を参照してきましたが、この米国標準をグローバル標準として適用することは困難であったり、矛盾が生じたり、あるいは混乱を招く可能性があります。
  - セクション 9.1 に準拠するためのより良い方法は Mozilla's Server Side TLS などのガイドを参照したり、 既知の適切な構成をつくり 、既存で最新の TLS 評価ツールを使用して望ましいレベルのセキュリティを確保することです。

V10 悪性コード

管理目標

コードが以下の上位要件を満たすことを確認します。

- アプリケーションの他の部分に影響が及ばないよう、悪性活動がセキュアな方法で適切に処理される
- time bomb や他の time based 攻撃に繋がる問題を作り込んでいない
- 悪性サイトや許可されていないサイトとの秘密の通信 (“phone home”)を行わない
- 攻撃者が制御可能なバックドア、イースターエッグ、サラミ攻撃、ルートキット、不正コードが作り込まれていない

悪性コードを完全に見つけることは不可能です。コードの中に悪性コードや不要な機能が含まれていないことを確認するために、最善の努力を払う必要があります。

V10.1 コード整合性

悪性コードに対する最良の防御策は、「信頼はするが、検証もする」ことです。 不正なコードや悪性コードを取り込むことは、多くの管轄で犯罪行為となっています。そして 悪性コードを無くすためにガイドラインを設ける必要があります。

リード開発者による定期的なコードチェック（特に時間、I/O、またはネットワーク機能にアクセスする可能性があるコード）を行う必要があります。

#	説明	L1	L2	L3	CWE
10.1.1	時間関数、危険なファイル操作、ネットワーク接続など、悪性コードを検出できるコード分析ツールが使用されている。			✓	749

V10.2 悪意コード検索

悪性コードが作り込まれることは極めてまれです。また検出は困難です。コードを 1 行 1 行レビューするのはロジックボムを見つける助けにはなるでしょうが、最も経験を積んだコードレビュー担当者をもってしても、存在すると分かっていても悪性コードを見つけることは容易ではありません。

このセクションに準拠するには、サードパーティのライブラリを含むソースコードに完全にアクセスする必要があります。

#	説明	L1	L2	L3	CWE
10.2.1	アプリケーションのソースコードおよびサードパーティライブラリに、不正な通信 (“phone home”) またはデータ収集機能が含まれていないようにする。このような機能がある場合は、データを収集する前にユーザの許可を受ける。		✓	✓	359
10.2.2	アプリケーションが連絡先、カメラ、マイク、ロケーションなど、プライバシー関連の機能に対して不要な権限または過剰な権限を要求しない。		✓	✓	272
10.2.3	アプリケーションのソースコードおよびサードパーティのライブラリにバックドアが含まれていない。（ハードコードされたアカウントや鍵、文書化されていないアカウントや鍵、コードの難読化、文書化されていないバイナリ BLOB、ルートキット、アンチデバッグ、危険なデバッグ機能、古い機能、隠された機能）			✓	507

#	説明	L1	L2	L3	CWE
10.2.4	日付と時刻関連の機能を検索して、アプリケーションのソースコードとサードパーティのライブラリに時限爆弾(time bomb)が含まれていない。			✓	511
10.2.5	アプリケーションのソースコードとサードパーティのライブラリに、サラミ攻撃、ロジックバイパス (logic bypasses)、ロジックボム (logic bombs) などの悪意のあるコードが含まれていない。			✓	511
10.2.6	アプリケーションのソースコードとサードパーティのライブラリにイースターエッグやその他の望ましくない機能が含まれていない。			✓	507

V10.3 アプリケーションの整合性

アプリケーションがデプロイされた後も、悪性コードが挿入される可能性があります。 アプリケーションは、信頼されていないソースからの未署名のコードの実行やサブドメインテイクオーバ（subdomain takeover）などの一般的な攻撃から自身を保護する必要があります。

このセクションに準拠するには運用上、持続的に進める必要があります。

#	説明	L1	L2	L3	CWE
10.3.1	アプリケーションにクライアントまたはサーバの自動更新機能がある場合は、安全なチャネルを経由して、デジタル署名がされていることを確認する。インストールまたは実行する前に、アップデートするコードのデジタル署名を検証する必要がある。	✓	✓	✓	16
10.3.2	アプリケーションで、コード署名やサブリソースの完全性などの保護が使用されている。加えて、信頼できない場所やインターネットから取得したモジュール、プラグイン、コード、ライブラリなどをロードまたは実行しない。	✓	✓	✓	353
10.3.3	期限切れドメイン名、期限切れ DNS ポインタまたは CNAME、パブリックソースコードリポジトリでの期限切れプロジェクト、一時的なクラウド API、サーバレス機能、ストレージバケット（autogen-bucketid.cloud.example.com）など、DNS エントリまたは DNS サブエントリに依存している場合、アプリケーションがサブドメイン奪取（subdomain takeover）から保護されている。アプリケーションの保護は使用する DNS 名の有効期限または変更を定期的にチェックすることを含めます。	✓	✓	✓	350

参考情報

- Hostile Subdomain Takeover, Detectify Labs
- Hijacking of abandoned subdomains part 2, Detectify Labs

V11 ビジネスロジック

管理目標

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ことを確認します。

- ビジネスロジックが正しい順序で処理
- ビジネスロジックに自動攻撃を検知し防止する制限が実装されている。自動攻撃の例としては，連続的な少額の送金や 1 度に 100 万人の友人を追加する，などがある
- 高い価値を持つビジネスロジック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 (spoofing)，改ざん (tampering)，情報の漏えい (information disclosure)，権限昇格 (elevation of privilege) 攻撃の対策を行っている

V11.1 ビジネスロジックのセキュリティ

ビジネスロジックセキュリティは、すべてのアプリケーションごとに固有なため、どのチェックリストも適用はできません。そして外部からの脅威から保護するように設計されている必要がありますが、それは Web アプリケーションファイアウォール (firewalls) や安全な通信を使用しても保護することはできません。OWASP Cornucopia などのツールを使用して、デザインスプリント中に脅威モデリングを使用することを推奨します。

#	説明	L1	L2	L3	CWE
11.1.1	アプリケーションが同じユーザのビジネスロジックフローを手順通り、省略せずに処理する。	✓	✓	✓	841
11.1.2	アプリケーションは、すべてのステップが人間の実際に処理する時間で処理されたビジネスロジックフローのみ処理する。（送信されるのが早すぎるトランザクションは処理されない。）	✓	✓	✓	799
11.1.3	アプリケーションに適切な制限が実装されており、特定のビジネス活動やトランザクションに対し、ユーザごとに適用されている。	✓	✓	✓	770
11.1.4	大量のデータの流出、ビジネスロジック要求、ファイルのアップロード、DoS 攻撃 (denial of service attacks) などの過剰な呼び出しから保護するために、アプリケーションに自動攻撃を防止する制限がある。	✓	✓	✓	770
11.1.5	アプリケーションは脅威モデリング (threat modeling) または類似の方法を使用して特定されたビジネスリスクから保護するために、ビジネスロジックの制限またはバリデーションがある。	✓	✓	✓	841
11.1.6	アプリケーションの機密性の高い処理が「Time Of Check to Time Of Use」TOCTOU 問題や、その他の競合状態の影響を受けない。		✓	✓	367
11.1.7	アプリケーションはビジネスロジックの観点から異常なイベントまたはアクティビティを監視している。例えば、順序がおかしい行為や通常のユーザが決して試みない行為の試行が該当する。 (C9)		✓	✓	754
11.1.8	アプリケーションは、自動攻撃または異常なアクティビティが検出されたときにアラートする設定機能がある。		✓	✓	390

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 4.1: Business Logic Testing
- 対自動処理はこれらを含む、多くの方法で対応できる。 OWASP AppSensor , OWASP Automated Threats to Web Applications
- OWASP AppSensor は、攻撃検知と対応に役立つ。
- OWASP Cornucopia

V12 ファイルとリソース

管理目標

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ことを確認します。

- 信頼できないファイルのデータがセキュアな方法で適切に処理される
- 信頼できない情報源から取得したデータは，Web ルート(webroot) の外に保存され，アクセスが制限される

V12.1 ファイルアップロード

高圧縮ファイル爆弾（zip bombs）はペネトレーションテスト技法を用いて充分テスト可能ですが、慎重な手動テストによる設計と開発の配慮を奨励し、かつ自動または不十分な手動ペネトレーションテストによるサービス運用妨害状態（DoS）を回避するために、L2 以上と見なされます。

#	説明	L1	L2	L3	CWE
12.1.1	ストレージを圧迫させたり、DoS 攻撃を引き起こしたりする可能性のある大きなファイルをアプリケーションが受け付けない。	✓	✓	✓	400
12.1.2	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。		✓	✓	409
12.1.3	1 人のユーザがあまりにも多くのファイルまたは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。		✓	✓	770

V12.2 ファイルの完全性

#	説明	L1	L2	L3	CWE
12.2.1	信頼できない場所から取得したファイルが、期待されるファイルであることをファイルの内容に基づいて判断する。		✓	✓	434

V12.3 ファイル実行

#	説明	L1	L2	L3	CWE
12.3.1	ユーザが送信したファイル名のメタデータがシステムまたはフレームワークにより直接使用されていない。パストラバーサルから保護するために URL API が使用されている。	✓	✓	✓	22
12.3.2	ローカルファイル（LFI）の漏えい、作成、更新、または削除を防止するために、ユーザが送信したファイル名のメタデータをバリデートもしくは無視する。	✓	✓	✓	73
12.3.3	リモートファイルインクルージョン (Remote File Inclusion, RFI) またはサーバサイドリクエストフォージェリ (Server-side Request Forgery, SSRF) 攻撃によるリモートファイルの漏えいまたは実行を防ぐために、ユーザが送信したファイル名のメタデータをバリデートもしくは無視する。	✓	✓	✓	98

#	説明	L1	L2	L3	CWE
12.3.4	アプリケーションを反射型ファイルダウンロード（Reflective File Download, RFD）から保護するため、ユーザが送信したファイル名（JSON、JSONP または URL パラメータの中にある）をバリデートまたは無視し、レスポンスの Content-Type ヘッダは text/plain に設定、および ContentDisposition ヘッダは固定ファイル名を指定する。	✓	✓	✓	641
12.3.5	OS コマンドインジェクションから保護するために、信頼できないファイルメタデータをシステム API またはライブラリで直接使用しない。	✓	✓	✓	78
12.3.6	アプリケーションは、未検証のコンテンツ配信ネットワーク、JavaScript ライブラリ、ノード npm ライブラリ、サーバサイド DLL などの信頼できない場所からの機能を含まず、かつ実行されない。		✓	✓	829

V12.4 ファイル保存

#	説明	L1	L2	L3	CWE
12.4.1	信頼できない場所から取得したファイルは、Web ルート以外にパーミッションを制限した上で保存する。	✓	✓	✓	552
12.4.2	信頼できない場所から取得したファイルが、アプリケーションが想定する種類であることを検証し、既知の悪性コンテンツがアップロードおよび配信されるのを防止するためにアンチウイルスソフトで検査する。	✓	✓	✓	509

V12.5 ファイルダウンロード

#	説明	L1	L2	L3	CWE
12.5.1	意図しない情報やソースコードの漏えいを防ぐために、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 Web 層が構成されている。例えば、バックアップファイル（.bak など）、一時作業ファイル（.swp など）、圧縮ファイル（.zip、.tar.gz など）およびエディタで一般的に使用されるその他の拡張子は、必要ない限り処理しない。	✓	✓	✓	552
12.5.2	アップロードされたファイルへの直接のリクエストが HTML/JavaScript コンテンツとして実行されない。	✓	✓	✓	434

V12.6 SSRF からの保護

#	説明	L1	L2	L3	CWE
12.6.1	Web サーバまたはアプリケーションサーバが、リクエストを送信したりデータ/ファイルをロードしたりできるように、リソースまたはシステムに許可リストが構成されている。	✓	✓	✓	918

参考情報

詳しくは以下の情報を参照してください。

- File Extension Handling for Sensitive Information
- Reflective file download by Oren Hafif
- OWASP Third Party JavaScript Management Cheat Sheet

V13 API と Web サービス

管理目標

信頼されたサービスレイヤ API（一般的に JSON や XML、または GraphQL）を使用する検証対象のアプリケーションが以下を備えていることを確認します。

- すべての Web サービスで適切な認証、セッション管理および認可
- 低信頼レベルから高信頼レベルに移行する全てのパラメータの入力正当性確認
- クラウドやサーバレス API を含む全ての API の種類に対して有効なセキュリティ管理策の実施

この章を他の全ての章と組み合わせて同じレベルで読んでください。 認証や API セッション管理の課題については重複しません。

V13.1 一般的な Web サービスセキュリティ

#	説明	L1	L2	L3	CWE
13.1.1	SSRF 攻撃や RFI 攻撃で使用される可能性があるような、異なる URI またはファイルのパーサを悪用する攻撃を回避するために、すべてのアプリケーションコンポーネントが同じエンコードおよびパーサを使用する。	✓	✓	✓	116
13.1.2	[削除, 4.3.1 と重複]				
13.1.3	API URL が API Key やセッショントークンなどの機密情報を公開していない。	✓	✓	✓	598
13.1.4	認可の判定が、URI とリソースレベルの両方で行われている（URI ではコントローラまたはルータで実施するプログラム型または宣言型のセキュリティによって、リソースレベルではモデルベースの許可によって実施される）。		✓	✓	285
13.1.5	予期しないまたは欠落している Content Type を含むリクエストが、適切なヘッダ（HTTP レスポンスステータス 406 Unacceptable または 415 Unsupported Media Type）でリジェクトされる。		✓	✓	434

V13.2 RESTful Web サービス

JSON schema のバリデーションの標準化のドラフト段階にあります（参考情報を参照）。RESTful Web サービスのベストプラクティスである JSON schema バリデーションの使用を検討するときは、JSON schema バリデーションと組み合わせ以下の追加のデータ検証の戦略を使用することを検討してください。

- 不足している要素や余分な要素があるかなど、JSON オブジェクトの解析検証
- データタイプ、データ形式、長さなどの標準入力検証メソッドを使用した JSON オブジェクト値のバリデーション
- そして、正式な JSON schema のバリデーション

JSON schema のバリデーション標準が正式化されると、ASVS はこの領域でのアドバイスを更新します。使用中の JSON schema バリデーションのライブラリを注意深く監視します。標準が正式になり、バグがリファレンス実装から解決されるまで、定期的に更新する必要があります。

#	説明	L1	L2	L3	CWE
13.2.1	保護された API またはリソースに対して通常ユーザが DELETE または PUT を使用するのを防ぐなど、有効化されている RESTful HTTP メソッドが、ユーザまたはアクションにとって妥当となっている。	✓	✓	✓	650
13.2.2	JSON スキーマバリデーションが設定され、入力を受け付ける前に確認されている。	✓	✓	✓	20
13.2.3	Cookie を使用する RESTful Web サービスが、次のうち少なくとも 1 つ以上を使って、クロスサイトリクエストフォージェリから保護されている。三重または二重送信クッキーパターン、CSRF ナンス、Origin リクエストヘッダのチェック。	✓	✓	✓	352
13.2.4	[削除, 11.1.4 と重複]				
13.2.5	受信した Content-Type が application/xml や application/json などの予期されるものであることを、REST サービスが明示的に検査している。		✓	✓	436
13.2.6	メッセージヘッダとペイロードが信頼でき、転送中に変更されていない。機密性、完全性の保護のため、転送に強力な暗号化（TLS のみ）を要求することは、多くの場合これで十分です。メッセージごとのデジタル署名は、高度なセキュリティを必要とするアプリケーションに対して、転送保護に加えて追加の保証を提供することができますが、複雑さやリスクをもたらす可能性があります。		✓	✓	345

V13.3 SOAP Web サービス

#	説明	L1	L2	L3	CWE
13.3.1	適切に形成された XML 文書を確保するために、XSD スキーマバリデーションに続いて、そのデータの処理が行われる前に各入力フィールドのバリデーションが実施されている。	✓	✓	✓	20
13.3.2	クライアントとサービス間の信頼できる転送を確保するために、メッセージペイロードが WS-Security を使用して署名されている。		✓	✓	345

注：DTD に対する XXE 攻撃の問題があるため、DTD 検証は使用しないでください。また、V14 構成で設定された要件に従って、フレームワーク DTD 評価を無効にしてください。

V13.4 GraphQL

#	説明	L1	L2	L3	CWE
13.4.1	高コストで、ネストされたクエリの結果として GraphQL またはデータレイヤエクスプレッションがサービス運用妨害（DoS）となることを防止するために、クエリ許可リストまたは、Depth 制限と量の制限の組み合わせを使用している。より高度なシナリオでは、クエリコスト分析を使用する必要があります。		✓	✓	770
13.4.2	GraphQL または他のデータレイヤの認可ロジックが、GraphQL レイヤではなくビジネスロジックレイヤに実装されている。		✓	✓	285

参考情報

詳しくは以下の情報を参照してください。

- OWASP Serverless Top 10
- OWASP Serverless Project
- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing
- OWASP Cross-Site Request Forgery cheat sheet
- OWASP XML External Entity Prevention Cheat Sheet - General Guidance
- JSON Web Tokens (and Signing)
- REST Security Cheat Sheet
- JSON Schema
- XML DTD Entity Attacks
- Orange Tsai - A new era of SSRF Exploiting URL Parser In Trending Programming Languages

V14 構成

管理目標

検証対象のアプリケーションが以下を備えていることを確認します。

- 安全で、再現性があり、自動化可能なビルド環境
- サードパーティのライブラリ、依存関係、構成管理を強化し、アプリケーションに古いコンポーネントや安全でないコンポーネントが含まれないようにします

既定のアプリケーションを構成することは、インターネット上で安全である必要があり、安全に構成する必要があります。

V14.1 ビルドとデプロイ

ビルドパイプラインは再現性のあるセキュリティの基盤です。安全でない何かが発見されるたびに、それをソースコード、ビルドまたはデプロイスクリプトで解決し、自動的にテストが可能。既知のセキュリティの問題が本番環境に展開されないようにビルドを警告または中断する自動セキュリティおよび依存関係のチェックを備えたビルドパイプラインの使用を強く勧めます。不規則に実行される手動手順は、回避可能なセキュリティのミスに直結します。

業界として DevSecOps モデルに移行するにつれ、「Known good（既知の良好な）」状態を実現するには、デプロイメントと構成の継続的な可用性と整合性を確保することが重要。以前はシステムがハッキングされた場合に、それ以上の侵害が生じていないことを証明するのに数日から数か月かかりました。今日では、ソフトウェア定義のインフラストラクチャ、ダウンタイム無しでの迅速な A/B デプロイメント、そして自動化コンテナ化されたビルドでは、侵害された「既知の良好な」代替品を自動的かつ継続的にビルド、強化、デプロイすることができます。

もし従来のモデルがまだ存在する場合は、その構成を強化してバックアップするために手動の手順を実行し、侵害されたシステムを整合性の高い妥協のないシステムに迅速に交換できるようにする必要があります。

このセクションに準拠するには、自動ビルドシステムと、ビルドおよびデプロイスクリプトへのアクセスが必要です。

#	説明	L1	L2	L3	CWE
14.1.1	アプリケーションのビルドおよびデプロイプロセスが、CI / CD の自動化、自動構成管理、自動デプロイスクリプトなどの安全で再現性のある方法で実行されている。		✓	✓	
14.1.2	コンパイラフラグが、スタックのランダム化、データ実行防止などの利用可能なすべて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、または文字列操作が見つかった場合にビルドを中断するように構成されている。		✓	✓	120
14.1.3	使用しているアプリケーションサーバとフレームワークの推奨事項に従って、サーバ構成が強化されている。		✓	✓	16
14.1.4	アプリケーション、構成、およびすべての依存関係が、自動でデプロイスクリプトを使用して再度デプロイできるか、文書化およびテストされた Runbook から妥当な時間で構築できるか、またはバックアップからタイムリーに復元できる。		✓	✓	
14.1.5	許可された管理者が、セキュリティ関連のすべての構成の整合性を検証して改ざんを検出できる。			✓	

V14.2 依存関係

依存関係の管理は、あらゆる種類のあらゆるアプリケーションの安全な運用に欠かせません。 古くなった、または安全でない依存関係で最新の状態を維持できないことは、これまでで規模が大きく、とても高くつく攻撃の根本的な原因です。

注：レベル 1 では、14.2.1 の準拠は、より正確なビルド時の静的コード分析または依存関係分析ではなく、クライアント側およびその他のライブラリとコンポーネントの観察または検出に関係します。これらのより正確な手法は、必要に応じてインタビューによって発見できる場合があります。

#	説明	L1	L2	L3	CWE
14.2.1	すべてのコンポーネントが最新となっている。できればビルド時またはコンパイル時にディペンデンシチェッカを使用する。 (C2)	✓	✓	✓	1026
14.2.2	不要な機能、ドキュメント、サンプルアプリケーション、およびコンフィギュレーションがすべて削除されている。	✓	✓	✓	1002
14.2.3	JavaScript ライブラリ、CSS、Web フォントなどのアプリケーション資産がコンテンツ配信ネットワーク（Content Delivery Network, CDN）または外部プロバイダで外部的にホストされている場合、資産の整合性を検証するためにサブリソース完全性（SRI）が使用されている。	✓	✓	✓	829
14.2.4	サードパーティのコンポーネントが、事前に定義され、信頼され、継続的に維持されるリポジトリからのものとなっている。 (C2)		✓	✓	829
14.2.5	使用しているすべてのサードパーティライブラリのソフトウェア部品表 (SBOM) が維持されている。 (C2)		✓	✓	
14.2.6	サードパーティのライブラリをサンドボックス化またはカプセル化して、必要な動作だけをアプリケーションに公開することで、攻撃対象領域を最小化する。 (C2)		✓	✓	265

V14.3 意図しないセキュリティの開示

本番環境の構成を強化して、デバッグコンソールなどの一般的な攻撃から保護し、クロスサイトスクリプティング（Cross-site Scripting, XSS）とリモートファイルインクルード（Remote File Inclusion, RFI）攻撃の基準を引き上げ、歓迎されない多くのペネトレーションテストの報告の特徴である発見された些細な脆弱性を排除します。これらの問題の多くはめったに重大なリスクとして評価されませんが、他の脆弱性と連鎖します。これらの問題がデフォルトで存在しない場合、ほとんどの攻撃が成功する前にレベルが引き上げられます。

#	説明	L1	L2	L3	CWE
14.3.1	[削除, 7.4.1 と重複]				
14.3.2	Web またはアプリケーションサーバとアプリケーションフレームワークのデバッグモードが運用環境で無効になっていることを確認して、デバッグ機能、開発者コンソール、および意図しないセキュリティ開示を排除する。	✓	✓	✓	497
14.3.3	HTTP ヘッダまたは HTTP レスポンスの一部がシステム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。	✓	✓	✓	200

V14.4 HTTP セキュリティヘッダ

#	説明	L1	L2	L3	CWE
14.4.1	すべての HTTP レスポンスに Content-Type ヘッダが含まれている。またコンテンツタイプが text/*, /+xml および application/xml の場合には安全な文字セット (UTF-8, ISO-8859-1 など) を指定する。コンテンツは提供された Content-Type ヘッダと一致する必要がある。	✓	✓	✓	173
14.4.2	すべての API レスポンスに Content-Disposition:attachment; filename="api.json" ヘッダが含まれている（または他のコンテントタイプの適切なファイル名）。	✓	✓	✓	116
14.4.3	HTML、DOM、JSON、JavaScript インジェクションの脆弱性などの XSS 攻撃の影響を軽減するのに役立つ Content Security Policy (CSP) レスポンスヘッダが配置されている。	✓	✓	✓	1021
14.4.4	すべてのレスポンスに X-Content-Type-Options: nosniff ヘッダが含まれている。	✓	✓	✓	116
14.4.5	Strict-Transport-Security ヘッダがすべてのレスポンスとすべてのサブドメインに含まれている。例えば、Strict-Transport-Security：max-age=15724800; includeSubdomains	✓	✓	✓	523
14.4.6	URL 内の機密情報が Referer ヘッダを介して信頼できない関係者に公開されないように、適切な Referrer-Policy ヘッダが含まれている。	✓	✓	✓	116
14.4.7	Web アプリケーションのコンテンツはデフォルトでサードパーティのサイトに埋め込むことができない、および適切な Content-Security-Policy: frame-ancestors と X-Frame-Options レスポンスヘッダを使用して必要な場所でのみ正規のリソースの埋め込みが許可されている。	✓	✓	✓	1021

V14.5 HTTP リクエストヘッダのバリデーション

#	説明	L1	L2	L3	CWE
14.5.1	アプリケーションサーバが、pre-flight OPTIONS を含む、アプリケーションまたは API で使用されている HTTP メソッドのみを受け入れる。アプリケーションコンテキストに対する無効なリクエストについてログ出力やアラート発行する。	✓	✓	✓	749
14.5.2	提供された Origin ヘッダは、攻撃者によって簡単に変更できるため、認証やアクセス制御の判断に使用されていない。	✓	✓	✓	346
14.5.3	オリジン間リソース共有（Cross-Origin Resource Sharing, CORS）の Access-Control-Allow-Origin ヘッダが信頼できるドメインの厳密な許可リストを使用して照合し、「null」オリジンをサポートしていない。	✓	✓	✓	346
14.5.4	信頼できるプロキシまたは bearer トークンのような SSO デバイスによって追加された HTTP ヘッダがアプリケーションによって認証されている。		✓	✓	306

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 4.1: Testing for HTTP Verb Tampering
- Content-Disposition の API レスポンスへの追加はクライアントとサーバ間の MIME タイプの誤認識におよびファイル名オプションによる多くの攻撃から防御するのに役立つ。 Reflected File Download attacks.
- Content Security Policy Cheat Sheet
- Exploiting CORS misconfiguration for BitCoins and Bounties
- OWASP Web Security Testing Guide 4.1: Configuration and Deployment Management Testing
- Sandboxing third party components

Appendix A: 用語集

- **アドレス空間配置のランダム化 (Address Space Layout Randomization)** (ASLR) – メモリ破損のバグ悪用をより困難にする手法
- **許可リスト (Allow list)** – アプリケーションにおいて許可されているデータまたは操作のリスト (例：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)
- **アプリケーションセキュリティ (Application Security)** – OS やネットワークではなく，OSI 参照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重点をおく、アプリケーションレベルのセキュリティ
- **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
- **アプリケーションセキュリティ検証報告書 (Application Security Verification Report)** – 対象となるアプリケーションの分析と検証結果をまとめた報告書
- **認証 (Authentication)** – アプリケーションのユーザが正当であることの検証
- **自動検証 (Automated Verification)** – シグネチャを使って脆弱性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)を用いた検査
- **ブラックボックステスト (Black box testing)** – アプリケーションの内部構造や仕組みを調べずにアプリケーションの機能を調べるソフトウェアテストの方法
- **コンポーネント (Component)** – 独立したコード単位。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ーフェイスとの関連をもちます
- **クロスサイトスクリプティング (Cross-Site Scripting)** (XSS) – クライアントからコンテンツへのスクリプトの注入を可能にする、Web アプリケーションに典型的なセキュリティ上の脆弱性
- **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、ハードウェアやソフトウェア、ファームウェア
- **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の弱点一覧。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・軽減・防止の取り組みのベースラインとして機能します
- **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
- **動的アプリケーションセキュリティテスト (Dynamic Application Security Testing)** (DAST) - 技術は、実行状態にあるアプリケーションのセキュリティの脆弱性を示す状況を検出するように設計されています
- **動的検証 (Dynamic Verification)** – アプリケーションの実行中に脆弱性のシグネチャを用いて問題点を検出する自動化ツールを使用すること
- **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
- **グローバル一意識別子 (Globally Unique Identifier)** (GUID) – 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
- **Hyper Text Transfer Protocol** (HTTPS) – 分散、コラボレーション、ハイパーメディア情報システム用のアプリケーションプロトコル．World Wide Web におけるデータ通信の基盤
- **ハードコードされた鍵 (Hardcoded keys)** – コード、コメント、ファイルなど、ファイルシステムに格納されている暗号化鍵

- **ハードウェアセキュリティモジュール (Hardware Security Module)** (HSM) - 暗号化鍵とその他のシークレットを保護された方法で保存できるハードウェアコンポーネント
- **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
- **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
- **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オーナ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セキュリティポリシーを回避します。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
- **Open Web Application Security Project** (OWASP) – The Open Web Application Security Project (OWASP) は、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、自由でオープンなコミュニティ。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、アプリケーションセキュリティのリスクについて十分な情報に基づいた決断を下すことができることです。https://www.owasp.org/ を参照
- **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
- **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
- **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。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
- **個人を特定できる情報 (Personally Identifiable Information)** (PII) - 単独または他の情報と共に使用して、1 人の人物を識別したり、連絡したり、居場所を特定したり、または状況に応じて個人を識別したりできる情報
- **位置独立実行形式 (Position-independent executable)** (PIE) - 1 次メモリのどこかに配置されて、絶対アドレスに関係なく適切に実行される機械語
- **公開鍵暗号基盤 (Public Key Infrastructure)** (PKI) - 公開鍵をエンティティのそれぞれの ID にバインドする取り決めです。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
- **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
- **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。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
- **静的アプリケーションセキュリティテスト (Static application security testing)** (SAST) - アプリケーションのソースコード、バイトコード、セキュリティの脆弱性を示すコーディングおよびバイナリを分析するために設計された一連の技術。SAST ソリューションは、実行されていない状態の「インサイドアウト」からアプリケーションを分析
- **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

- **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。どこでどのようにセキュリティ管理策を使用しているか、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記述します
- **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の管理を左右するランタイム設定
- **セキュリティ管理 (Security Control)** – セキュリティチェック(例:アクセス制御の検査など)を実行する、あるいは呼出し結果がセキュリティに影響を与えうる(例：監査レコードの生成など)、機能や構成要素
- **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 を提供または変更することにより、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
- **シングルサインオン認証 (Single Sign-on Authentication)** (SSO) - ユーザが 1 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。例えば、Google にログインすると、YouTube、Google Docs、Gmail などの他の Google サービスにアクセスすると、自動的にログインします
- **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の 1 つ。悪性 SQL 文がデータの入力箇所に挿入されます
- **SVG** - Scalable Vector Graphics
- **タイムベース OTP (Time-based OTP)** - 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
- **脅威モデリング (Threat Modeling)** - 脅威の主体、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
- **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
- **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
- **二要素認証 (Two-factor authentication)** (2FA) - アカウントログインに第二レベルの認証を追加します
- **Universal 2nd Factor** (U2F) - USB または NFC セキュリティキーを二番目の認証要素として使用できるようにするために FIDO により作成された標準の一つ
- **URI/URL/URL フラグメント (URI/URL/URL fragments)** – Uniform Resource Identifier (URI) は、名前または Web リソースを識別するために使用される文字列。多くの場合リソースへの参照として使用されます
- **検証者 (Verifier)** – OWASP ASVS の要件に基づいてアプリケーションをレビューする個人またはグループ
- **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
- **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 公開鍵暗号基盤（PKI）標準を使用して、公開鍵が証明書に含まれるユーザやコンピュータまたはサービス ID に属していることを確認するデジタル証明書
- **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。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

Appendix 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/
2. OWASP Web Security Testing Guide: https://owasp.org/www-project-web-security-testing-guide/
3. OWASP Proactive Controls: https://owasp.org/www-project-proactive-controls/
4. OWASP Security Knowledge Framework: https://owasp.org/www-project-security-knowledge-framework/
5. OWASP Software Assurance Maturity Model (SAMM): https://owasp.org/www-project-samm/

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 には ASVS のさまざまなトピックに関連する多数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/cheatsheets/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/
2. OWASP Mobile Top 10 Risks: 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/

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/

OWASP Serverless プロジェクト

1. OWASP Serverless Project: 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
2. MITRE Common Weakness Enumeration: https://cwe.mitre.org/
3. PCI Security Standards Council: 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
5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures: 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

Appendix C: Internet of Things の検証要件

この章は元々メインブランチにありましたが、OWASP IoT チームが行った作業によって、このテーマに関する 2 つの異なる基準を維持することは意味がありません。4.0 リリースでは、これを付録に移動し、これを必要とするすべての人に、メインの OWASP IoT project を使用することを推奨します。

管理目標

組み込み/IoT 機器は、

- 信頼できる環境でセキュリティ管理を実施することによって、サーバ内と同じレベルのセキュリティ管理をデバイス内でも行う
- デバイスに保存されている機密データは、セキュアエレメントなどのハードウェアに保護されたストレージを使用して、安全な方法で実行する必要があります
- デバイスから送信されるすべての機密データは、TLS を利用する必要があります

セキュリティ検証要件

#	説明	L1	L2	L3	Since
C.1	USB、UART、そして他のシリアルバリアントのような、アプリケーション層のデバッグインターフェイスが無効になっているか、複雑なパスワードによって保護されている。	✓	✓	✓	4.0
C.2	暗号鍵と証明書が各デバイスに固有となっている。	✓	✓	✓	4.0
C.3	ASLR や DEP などのメモリ保護制御が、組み込み/ IoT オペレーティングシステムによって有効になっている（該当する場合）。	✓	✓	✓	4.0
C.4	JTAG や SWD などのオンチップデバッグインターフェイスが無効になっている、または使用可能な保護メカニズムが有効になっていて適切に構成されている。	✓	✓	✓	4.0
C.5	SoC デバイスまたは CPU で利用可能な場合、Trusted Execution が実装および有効になっている。	✓	✓	✓	4.0
C.6	機密データ、秘密鍵、そして証明書が、セキュアエレメント、TPM、TEE（Trusted Execution Environment）に安全に保存されていること、または強力な暗号化を使用して保護されている。	✓	✓	✓	4.0
C.7	ファームウェアアプリが Transport Layer Security（トランスポートレイヤセキュリティ）を使用して伝送中のデータを保護している。	✓	✓	✓	4.0
C.8	ファームウェアアプリがサーバ接続のデジタル署名を検証する。	✓	✓	✓	4.0
C.9	ワイヤレス通信が相互に認証されている。	✓	✓	✓	4.0
C.10	ワイヤレス通信が暗号化されたチャネルを介して送信される。	✓	✓	✓	4.0
C.11	禁止された C 関数の使用が適切な安全な同等の関数に置き換えられている。	✓	✓	✓	4.0
C.12	各ファームウェアがサードパーティのコンポーネント、バージョン、および公開された脆弱性をカタログ化するソフトウェア部品表を維持している。	✓	✓	✓	4.0

#	説明	L1	L2	L3	Since
C.13	サードパーティのバイナリ、ライブラリ、フレームワークを含むすべてのコードがハードコードされたクレデンシャル（バックドア）についてレビューされている。	✓	✓	✓	4.0
C.14	シェルコマンドラッパー、スクリプトを呼び出して、アプリケーションおよびファームウェアコンポーネントが OS コマンドインジェクションの影響を受けないこと、またはセキュリティ制御により OS コマンドインジェクションが阻止されている。	✓	✓	✓	4.0
C.15	ファームウェアのアプリが信頼できるサーバを固定化している。		✓	✓	4.0
C.16	耐タンパ性または、改ざん検知機能が存在する。		✓	✓	4.0
C.17	チップ製造者による、有用な知的財産権保護技術が有効化されている。		✓	✓	4.0
C.18	ファームウェアのリバースエンジニアリングを防止するためのセキュリティ管理策（デバッグ情報の除去など）が実施されている。		✓	✓	4.0
C.19	デバイスが、ブートイメージのロード前に署名の検証を行う。		✓	✓	4.0
C.20	ファームウェアの更新処理が TOCTOU 攻撃に対して脆弱でない。		✓	✓	4.0
C.21	デバイスがコード署名を利用し、インストール前にファームウェアのアップグレードファイルの検証を行う。		✓	✓	4.0
C.22	デバイスが、古いバージョンのファームウェアにダウングレードしない。		✓	✓	4.0
C.23	暗号論的に安全な擬似乱数生成器が、組込み機器上で使用されていること（例えば、チップが提供する乱数生成器を使用している等）。		✓	✓	4.0
C.24	ファームウェアが事前設定のスケジュールに従って、ファームウェアの自動更新を実行する。		✓	✓	4.0
C.25	改ざんを検知、不正なメッセージを受信した際に、デバイスがファームウェアおよび機密データをワイプする。			✓	4.0
C.26	デバッグ用インターフェース（JTAG/SWD など）を無効化できるマイクロコントローラだけが使用されている。			✓	4.0
C.27	デキャップやサイドチャネル攻撃から防御できるマイクロコントローラを使用する。			✓	4.0
C.28	機微な痕跡がプリント基板の外部レイヤに漏えいしない。			✓	4.0
C.29	チップ間の通信（メインボードからドーターボードへの通信など）を暗号化している。			✓	4.0
C.30	デバイスがコード署名を使用し、実行前にコードの妥当性の検証を行う。			✓	4.0
C.31	メモリ内に保持される機密な情報が、不要になったら直ちにゼロで上書きされる。			✓	4.0

#	説明	L1	L2	L3	Since
C.32	ファームウェアアプリがアプリ分離のため、カーネルコンテナを利用している。			✓	4.0
C.33	ファームウェアが -fPIE、-fstack-protector-all、-Wl、-z、noexecstack、-Wl、-z、noexecheap などの安全なコンパイルフラグでビルドされている。			✓	4.0
C.34	（該当する場合）マイクロコントローラが、コードプロテクションを利用している。			✓	4.0

参考情報

詳しくは以下の情報を参照してください。

- OWASP Internet of Things Top 10
- OWASP Embedded Application Security Project
- OWASP Internet of Things Project
- Trudy TCP Proxy Tool