Application Security Verification Standard 3.0

2015 年 10 月

目次

- 目次
- 謝辞
  - バージョン 3.0 (2015 年)
  - バージョン 2.0 (2014 年)
  - バージョン 1.0 (2009 年)
- この標準について
- 著作権とライセンス
- 序文
  - 3.0 の更新内容
- Application Security Verification Standard の使用
  - アプリケーションセキュリティ検査レベル
    - この標準の使用方法
    - レベル 1: 初級
    - レベル 2: 標準
    - レベル 3: 上級
  - ASVS 適用の実際
- ケーススタディ
  - ケーススタディ 1: セキュリティテストガイドとして
  - ケーススタディ 2: セキュア SDLC として

ソフトウェアが検査レベルを満たすことの評価

ASVS 認定と信頼マークに関する OWASP の立ち位置

組織の認定の指針

自動ペネトレーションテストツールの役割

ペネトレーションテストの役割

セキュリティアーキテクチャの詳細な指針として

既製のセキュアコーディングチェックリストに代わるものとして

自動単体および統合テストのガイドとして

セキュア開発のトレーニングとして

ASVS を使用する OWASP プロジェクト

Security Knowledge Framework

OWASP Zed Attack Proxy

OWASP Cornucopia

詳細な検査要件

V1: アーキテクチャ、設計、脅威モデリング

管理目標

要件

参照先

V2: 認証に関する検査要件

管理目標

要件

参照先

V3: セッション管理に関する検査要件

管理目標

要件

参照先

V4: アクセス制御に関する検査要件

管理目標

要件

参照先

V5: 悪性入力の処理に関する検査要件

管理目標

要件

参照先

V6: 出力のエンコード / エスケープ

V7: 保存データの暗号化に関する検査要件

管理目標

要件

参照先

V8: エラー処理とログの保存に関する検査要件

管理目標

要件

参照先

V9: データの保護に関する検査要件

管理目標

要件

参照先

V10: 通信セキュリティに関する検査要件

管理目標

要件

参照先

V11: HTTP のセキュリティ設定に関する検査要件

管理目標

要件

参照先

V12: セキュリティ設定に関する検査要件

V13: 悪性活動の適切な処理に関する検査要件

管理目標

要件

V14: 内部セキュリティに関する検査要件

V15: ビジネスロジックに関する検査要件

管理目標

要件

参照先

V16: ファイルとリソースに関する検査要件

管理目標

要件

参照先

V17: モバイルに関する検査要件

管理目標

要件

参照先

V18: Web サービスに関する検査要件

管理目標

要件

参照先

V19: 構成

管理目標

要件

参照先

付録 A: 廃止された要件に関する追跡情報

付録 B: 用語集

付録 C: 参照先

付録 D: 標準の対応

謝辞

バージョン 3.0 (2015 年)

プロジェクトリーダー	主要作成者	共同作成者およびレビュー担当者
Andrew van der Stock Daniel Cuthbert	Jim Manico	Boy Baukema Ari Kesäniemi Colin Watson François-Eric Guyomarc'h Cristinel Dumitru James Holland Gary Robinson Stephen de Vries Glenn Ten Cate Riccardo Ten Cate Martin Knobloch Abhinav Sejpal David Ryan Steven van der Baan Ryan Dewhurst Raoul Endres Roberto Martelloni

バージョン 2.0 (2014 年)

プロジェクトリーダー	主要作成者	共同作成者およびレビュー担当者
Daniel Cuthbert Sahba Kazerooni	Andrew van der Stock Krishna Raja	Antonio Fontes Colin Watson Jeff Sergeant Pekka Sillanpää Archangel Cuison Dr Emin Tatli Jerome Athias Safuat Hamdy Ari Kesäniemi Etienne Stalmans Jim Manico Scott Luc Boy Baukema Evan Gaustad Mait Peekma Sebastien Deleersnyder

バージョン 1.0 (2009 年)

プロジェクトリーダー	主要作成者	共同作成者およびレビュー担当者
Mike Boberski Jeff Williams Dave Wichers		Andrew van der Stock Dr. Sarbari Gupta John Steven Pierre Parrend Barry Boyd Dr. Thomas Braun Ken Huang Richard Campbell Bedirhan Urgun Eoin Keary Ketan Dilipkumar Vyas Scott Matsumoto Colin Watson Gaurang Shah Liz Fong Shouvik Bardhan Dan Cornell George Lawless Mandeep Khera Stan Wisseman Dave Hausladen Jeff LoSapio Matt Presson Stephen de Vries Theodore Winograd Jeremiah Grossman Nam Nguyen Steve Coyle Dave van Stein John Martin Paul Douthit Terrie Diaz

この標準について

Application Security Verification Standard (アプリケーションセキュリティ検査標準) とは、アプリケーションのセキュリティ要件またはセキュリティテストの項目です。アプリケーションの設計担当者、開発者、テスト担当者、セキュリティプロフェッショナル、ユーザーは、この標準を使用してセキュアなアプリケーションを定義することができます。

著作権とライセンス

Copyright © 2008 – 2015 The OWASP Foundation. 本書は、クリエイティブコモンズ表示−継承 3.0 ライセンスに基づいて公開されています。再使用または頒布する場合は、他者に対して本著作物のライセンス条項を明らかにする必要があります。

序文

Application Security Verification Standard (ASVS) バージョン 3.0 にようこそ。ASVS は、最新の Web アプリケーションを設計、開発、テストするときに必要となる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ことを目指した、コミュニティによる取組みであり、特に、機能的および非機能的なセキュリティ管理を標準化することに重点をおいています。

ASVS v3.0 は、コミュニティによる取組みと業界からのフィードバックの成果です。この版では、ASVS を採用しての実際の使用例を具体的に示すことが重要であると考えました。そうすることで、ASVS に初めて接する企業がこの標準の採用計画を容易に行えるようになり、またこの標準を既に採用している企業は他の企業の経験から学ぶことができます。

この標準の内容に 100% 納得いただけるとは我々も考えていません。リスク分析は常にある程度主観的になるため、あらゆる対象に適用可能な標準として一般化することには困難が伴います。しかし、我々は、この版で行った最新の更新が、正しい方向へ踏み出す一歩となること、そして、この重要な業界標準に導入された概念を、尊重しつつ、さらに強化するものとなることを期待しています。

3.0 の更新内容

バージョン 3.0 では、構成、Web サービス、最新の (クライアント) ベースのアプリケーションなどの新たなセクションをいくつか追加し、最新のアプリケーションへのこの標準の適用可能性を強化するよう務めました。最新のアプリケーションは、一般に応答性を備えるアプリケーションであり、大規模な HTML5 フロントエンドあるいは、モバイルクライアントが SAML 認証を使用して共通の RESTful Web サービス群を呼び出します。

また、たとえば、モバイル開発者が同じ項目を何度も再テストしなくて済むように、標準内の重複を排除しました。

CWE 共通脆弱性一覧 (Common Weakness Enumeration: CWE) との対応表を作成しました。CWE 対応表を活用することで、脆弱性の悪用されやすさや悪用が成功した場合の影響を明らかにすることができます。平たく言えば、セキュリティ管理策が用いられていない、あるいは効果的に実装されていない場合にどのような問題が発生するか、また、脆弱性をどのように軽減できるかについての洞察を得る助けとなるということです。

また我々は、コミュニティに働きかけ、AppSec EU 2015 ではピアレビューセッションを、AppSec USA 2015 では最終的なワーキングセッションを行い、コミュニティから多くのフィードバックを取り込みました。ピアレビュー時には、管理策の意味に対する編集内容が大幅に変更された場合は、新たな管理策を作成し、以前のものを廃止しました。廃止した管理策については、混乱を招く可能性があるため、あえて再利用しないことにしました。変更内容については、付録 A に包括的な対応表を収録しました。

まとめると、v3.0 は、この標準の策定以来、きわめて大規模な変更が行われた唯一のバージョンとなっています。この標準の更新が皆様にとって有益なものとなり、さまざまな方法で活用されることを期待しています。

Application Security Verification Standard の使用

ASVS には、次の 2 つの大きな目的があります。

1. 組織におけるセキュアなアプリケーションの開発と保守を支援する
2. セキュリティサービス、セキュリティツールのベンダー、およびユーザーが、自身の要件と提供されるものを調整できるようにする

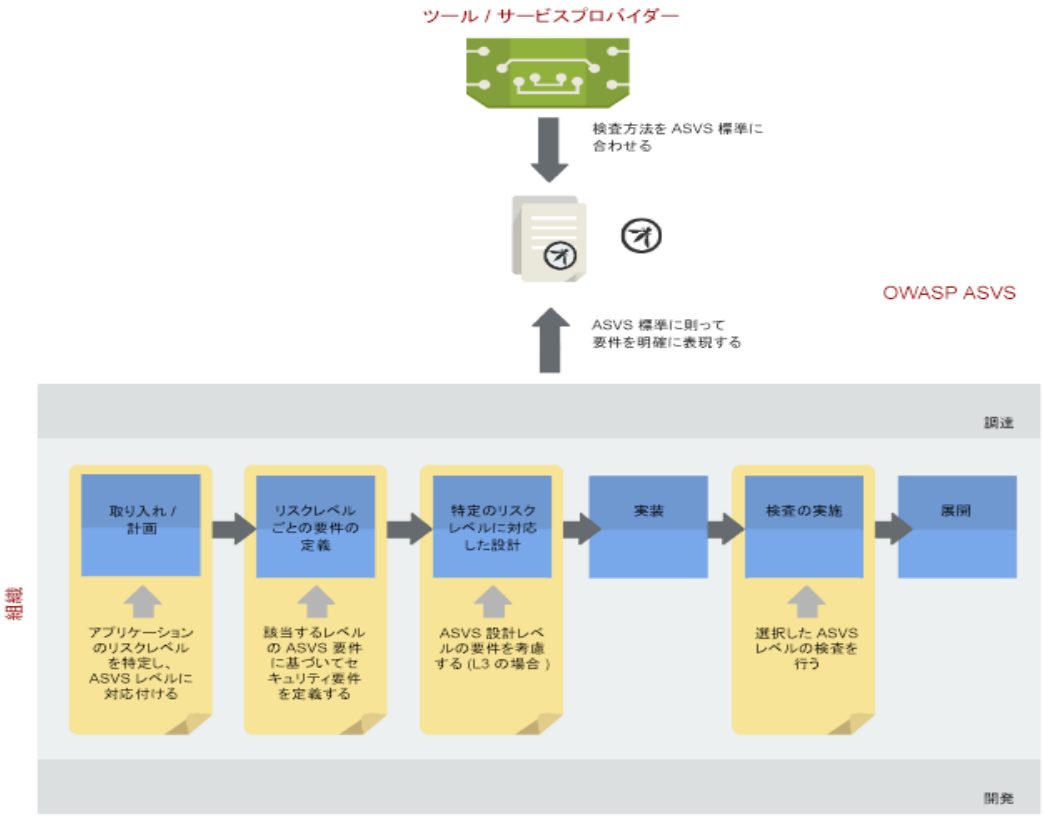

図 1: 組織およびツール / サービスプロバイダーによる ASVS の活用

アプリケーションセキュリティ検査レベル

Application Security Verification Standard では、3 段階のセキュリティ検査レベルを定義しており、レベルごとに深度が増します。

ASVS レベル 1 は、すべてのソフトウェアを対象としています。
ASVS レベル 2 は、保護を必要とする機密データを含むアプリケーション用です。
ASVS レベル 3 は、重要度が最も高いアプリケーション、つまり、最高レベルの信頼性を必要とするあらゆるアプリケーション (高い価値を伴う取引を行うものや、機密の医療データを含むものなど) を対象としています。

各 ASVS レベルには、セキュリティ要件のリストが含まれます。また、個々の要件は、開発者によってソフトウェアに組み込まれるべきセキュリティ上の機能に対応付けることができます。

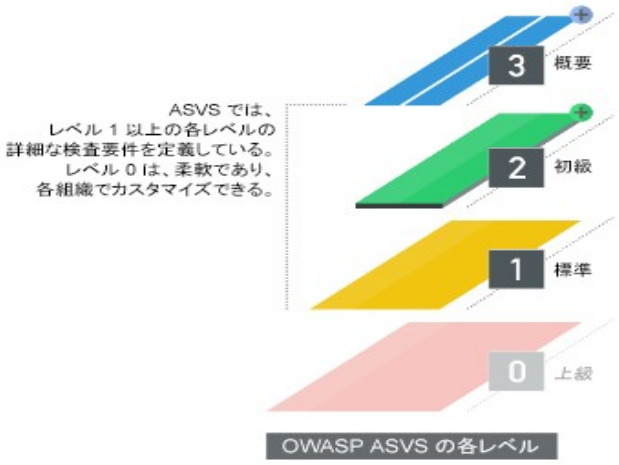

図 2: OWASP ASVS の各レベル

この標準の使用方法

Application Security Verification Standard の最適な使用方法の 1 つは、自社のアプリケーション、プラットフォーム、または組織固有のセキュアコーディングチェックリスト作成の雛形として使用することです。ユース

ケースに合わせて ASVS を調整することで、プロジェクトや環境において最も重要なセキュリティ要件により注力できるようになります。

レベル 1: 初級

OWASP Top 10 や他の同様のチェックリストに含まれる容易に検出可能な脆弱性に対して、防御が適切に適用されているアプリケーションは、ASVS レベル 1 (初級) を満たすと見なされます。

通常、レベル 1 は、セキュリティ管理策を正しく使用しているが、それほど厳密さが求められないアプリケーションに適しています。あるいは、エンタープライズアプリケーション群の簡易分析を行うためや、多段階の取組みの一環として行われるセキュリティ要件の優先順位リストの作成を補助する目的に適用しています。レベル 1 の管理策は、ツールを使用して自動的に確認することもできれば、単純に手動で (ソースコードにアクセスすることなく) 確認することもできます。レベル 1 は、すべてのアプリケーションが最低限満たすべきレベルであると考えています。

アプリケーションに対する脅威は、多くの場合、攻撃者が単純で容易に使える技術を用い、発見し悪用するのが容易な脆弱性を見つけることに端を発するものです。これとは対照的に、高いスキルと明確な動機を持つ攻撃者は、全エネルギーを集中させて特定のアプリケーションに狙いを定めてきます。アプリケーションが価値の高い重要なデータを処理する場合、おそらくレベル 1 のレビューで止まることはないでしょう。

レベル 2: 標準

今日のソフトウェアが直面するほとんどのリスクに対する防御が適切に適用されているアプリケーションは、ASVS レベル 2 (標準) を満たすと見なされます。

レベル 2 では、アプリケーション内にセキュリティ管理策が存在し、有効であり、活用されていることを保証します。通常、レベル 2 は、企業間のトランザクションを大量に処理するアプリケーション (医療情報を処理する、事業上重要もしくは機密の機能を実装する、その他の機密の資産を処理するなど) に適用されます。

レベル 2 アプリケーションに対する脅威は、高いスキルと明確な動機を持つ攻撃者によるものであり、彼らはアプリケーションの脆弱性の検出と攻略に定評のあるツールや技法を使用し、特定の標的に対して攻撃を行います。

レベル 3: 上級

ASVS レベル 3 は最も高い検査レベルです。通常、レベル 3 は、軍事、保健と安全、重要インフラストラクチャなどの分野で使われる、高度なセキュリティ検査レベルを必要とするアプリケーションを対象としています。障害の発生が組織の運営、さらにはその存続に大きな影響を及ぼしうる重要な機能を担うアプリケーションに対しては、ASVS レベル 3 が必要となるでしょう。ASVS レベル 3 の適用に関するいくつかの指針をこの後に示します。アプリケーションにおいて、高度な脆弱性に対する対策が適切に行われており、正しいセキュリティ設計の原則を実証しているアプリケーションは、ASVS レベル 3 (上級) を満たすと見なされます。

ASVS レベル 3 のアプリケーションには、他のどのレベルよりも綿密な分析、アーキテクチャ、コーディング、およびテストが必要になります。セキュアなアプリケーションは、目的を持ってモジュール化されており (弾力性、スケーラビリティ、また特に重要なものとして多層的セキュリティを実現するために)、各モジュール (ネットワーク接続によって、あるいは物理的に分離された) は、自身のセキュリティ上の役割を果たします (深層防御)。この役割については、適切に文書化される必要があります。セキュリティ上の役割には、機密性の確保 (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 (負荷の正常な処理など)、認証 (システム間の認証など)、否認不可、認可、および監査 (ログ) があります。

ASVS 適用の実際

脅威が異なればその動機も異なります。業界の中には、固有の情報資産や技術資産を持ち、その分野における法令遵守要件が適用されるものがあります。

以下に、業界別の推奨される ASVS レベルに関する指針を示します。業界における脅威にはそれぞれの独自の基準があり種類も異なるかもしれませんが、すべての業界に共通する問題があります。それは、初級レベルの攻撃者は容易に悪用できる脆弱なアプリケーションを探す、ということです。ASVS レベル 1 が業界を問わずすべてのアプリケーションに推奨するのは、このためです。レベル 1 は、リスク管理の出発点として、初級レベルの脅威に対処します。組織は、事業の性質に基づいて、自社固有のリスク特性をより綿密に調べる必要があります。この対極にあるのが ASVS レベル 3 です。レベル 3 は、生命や安全を脅かしかねない場合やアプリケーションが完全に侵害された際に組織へ甚大な影響が及ぼしうる場合に対応しています。

業界	脅威のプロファイル	ASVS L1 推奨対象	ASVS L2 推奨対象	ASVS L3 推奨対象
金融 / 保険	日和見的な攻撃者からの攻撃にもさらされますが、金銭詐取という明確な動機を持つ攻撃者が、高い価値を持つ標的を狙うことが多くなっています。攻撃者は通常、機密データや口座資格情報を探し、不正行為を働いたり、アプリケーション組込みの送金機能を使って直接的に利益を得ようとします。多くの場合、窃盗された認証情報の使用、アプリケーションに対する攻撃、ソーシャルエンジニアリングなどが攻撃手法です。 遵守すべき主要な法令として、PCI データセキュリティスタンダード (PCI DSS)、グラムリーチブライリー法、米国企業改革法 (SOX) などがあります。	ネットワークアクセスが可能なすべてのアプリケーション。	一定の金額を限定的な方法で移動することができる、クレジットカード番号や個人情報などの機密情報を含むアプリケーション。例として、次の場合があります。 (i) 同一機関の口座間の送金 (ii) 取引限度額がある、時間のかかる資金移動形式 (自動決済機関など) (iii) 一定期間内における送金額の限度額がある電信送金	大量の機密情報を含むアプリケーション、あるいは多額の迅速な送金 (電信送金など) や個別取引形式、少額送金の一括処理形式による多額の送金を実行できるアプリケーション。
製造、輸送、テクノロジ、公益事業、インフラ、防衛	これらの業界にはさほど共通点がないと思うかもしれませんが、これら業界の組織を狙う攻撃者は、より多くの時間と技術力とリソースを使って集中的な攻撃を行う傾向が強まっています。多くの場合、機密情報やシステムの特定は容易ではなく、内通者やソーシャルエンジニアリング技法の活用を必要とします。攻撃は、内通者、部外者、あ	ネットワークアクセスが可能なすべてのアプリケーション。	ソーシャルエンジニアリングに使われる可能性がある内部情報や従業員情報を含むアプリケーション。必須ではないが重要な知的財産または企業秘密を含むアプリケーション。	組織の存続や成功に不可欠な、高い価値を持つ知的財産、企業秘密、国家機密 (たとえば、米国の場合、"Secret" 以上のカテゴリに分類されるあらゆる情報) を含むアプリケーション。セキュリティに影響のある機能 (輸送、製造装置、制御システムなど) を制御するアプリケーション、または人命に関わるアプリケーション。

	るいはその両方の共謀によって行われる場合があります。攻撃者の目的には、知的財産にアクセスして戦略的または技術的な利益を得ることなどがあります。また、アプリケーションの機能を悪用して機密性の高いシステムの動作に影響を及ぼしたり、このようなシステムを中断させようとする攻撃者も見逃すわけにはいきません。 ほとんどの攻撃者は、個人識別情報や支払データなど直接的または間接的な利用によって利益につながる機密データを探しています。多くの場合、このようなデータは、なりすまし犯罪や不正支払いなど、さまざまな不正行為の計画に利用されます。			
医療	ほとんどの攻撃者は、直接的または間接的な利用によって利益を得ようと、個人識別情報や支払データなどの機密データを探しています。多くの場合、このようなデータは、なりすまし犯罪や不正支払いなど、さまざまな不正行為の計画に利用されます。 米国の医療業界の場合、医療保険の相互運用性と説明責任に関する法律 (HIPAA) のプライバシー、セキュリ	ネットワークアクセスが可能なすべてのアプリケーション。	センシティブな医療情報 (保護医療情報)、個人識別情報、または支払データを小・中規模含むアプリケーション。	人命に関わる医療装置、機器、または記録の制御に使用されるアプリケーション。詐欺に利用できるトランザクションデータを大量に含む支払システムや POS システム。これらのアプリケーションの管理インターフェイスも含まれます。

	ティ、侵害開示に関する規則と患者の安全性に関する規則 (http://www.hhs.gov/ocr/privacy/) があります。			
小売、食品、接客	行き当たりばったりに "力尽くで金品を奪う" のが、この分野の攻撃者によく見られる攻撃方法です。一方で、支払情報の保有、金融取引、個人識別情報の保存を行うアプリケーションに対する、特定の攻撃の脅威も存在します。前述の脅威よりも頻度は低いものの、業界を狙ったより高度な攻撃の可能性があり、知的財産の詐取、競合他社の情報収集、交渉中の組織やビジネスパートナーを出し抜くことを目的に行われます。	ネットワークアクセスが可能なすべてのアプリケーション。	ビジネスアプリケーション、製品カタログ情報、社内情報に適するほか、限定的なユーザー情報 (連絡先情報など) を含むアプリケーションに適します。小・中規模の支払データまたは精算機能を含むアプリケーション。	詐欺に利用されうる、トランザクションデータが大量に含まれる支払システムや POS システム。これらのアプリケーションの管理インターフェイスも含まれます。完全なクレジットカード番号、母親の旧姓、社会保障番号などの大量の機密情報が含まれるアプリケーション。

ケーススタディ

ケーススタディ 1: セキュリティテストガイドとして

米国ユタ州のある私立大学では、学内の Red Team が、アプリケーションペネトレーションテスト実施のガイドとして OWASP ASVS を使用しています。OWASP ASVS は初期段階の計画や対象を定めるミーティングからテストの実施指針まで、ペネトレーションテストのプロセス全体において、また、クライアント向けの最終報告書に検査結果を表現するための枠組みとしても使用されます。さらに、Red Team はチームのトレーニング計画にも ASVS を活用しています。

学内 Red Team は、大学全体の情報セキュリティ戦略の一環として、校内のさまざまな学部に対して、ネットワークとアプリケーションのペネトレーションテストを実施しています。初期の計画ミーティングの際、クライアントは多くの場合、学生チームによってアプリケーションがテストされることを許可したがりません。しかし、ASVS を紹介し、テストが ASVS 標準に則って行われ、最終報告書にはアプリケーションの標準に対する達成度合いが記されることを説明すると、ほとんどのクライアントは安心して任せてくれます。その後、ASVS はテスト範囲の検討に使用され、テストに要する時間と作業が見積もられます。Red Team は、ASVS のあらかじめ定義された検査レベルを使ってリスクに基づいたテストについて説明します。クライアント、関係者、チームは、ASVS を活用することでテスト対象となるアプリケーションに関する適切なスコープについて合意に至ります。

テストが開始すると、Red Team は ASVS を使って活動を計画し、作業量の分割を行います。チームのプロジェクトマネージャーは、各検査要件のテストの状況 (完了、保留中など) を追跡することによって、テストの進捗状況を容易に把握できます。これにより、プロジェクトマネージャーは、クライアントとよりスムーズにコミュニケーションを取れるようになり、またリソースをより適切に管理できるようになります。Red Team は主に学生から構成されているため、チームのほとんどのメンバーは複数の講義に時間を割かなくてはなりません。しかし、個々の検査要件またはカテゴリ全体に基づいてタスクが適切に定義されているため、チームメンバーは、テストすべき内容を正確に把握することができ、また個々のタスク完了までにかかる時間を正確に見積もることができます。ASVS は構成が明確であるため、報告書作成も簡単です。チームメンバーは、検査結果を記述してから次のタスクに進むことができ、ペネトレーションテストの進行に従って報告書の大半を効果的に作成できます。

Red Team は ASVS に沿って最終報告書を作成、各検査要件のステータスを報告し、必要に応じて詳細な情報を提供します。そのため、クライアントや関係者は、ASVS 標準に照らし合わせてアプリケーションの状況を適切に把握できます。また、最終報告書は、その後のセキュリティの向上や低下を知る上での目安になるので、そ

のあとの取組みにも非常に役立ちます。さらに、報告書形式は ASVS と密接に連携しているため、特定のカテゴリにおけるアプリケーションの達成度合いに関心がある関係者は、関連情報を容易に見つけ出すことができます。また、ASVS は構成が明確なため、新たなチームメンバーに報告書作成方法を教えることが以前の報告書形式よりも容易になりました。

また、ASVS を採用することで、Red Team のトレーニングは改善しています。以前は、チームリーダーまたはプロジェクトマネージャーが選択したトピックを中心とするトレーニングが週に 1 回実施されていました。トピックは、チームメンバーからのリクエストや、ニーズに基づいて選択されていました。このような基準に基づくトレーニングは、チームメンバーのスキルを広げる可能性があるとはいえ、Red Team の中心的な活動に必ずしも関連しているわけではありませんでした。つまり、ペネトレーションテストにおけるチームのスキルはそれほど向上していませんでした。ASVS の採用後、チームのトレーニングは、個々の検査要件のテスト方法に焦点を絞って実施されるようになりました。その結果、個々のチームメンバーの測定可能なスキルと最終報告書の品質が大幅に向上しました。

ケーススタディ 2: セキュアな SDLC として

金融機関向けにビッグデータ分析機能を提供しようとしているスタートアップ企業は、金融メタデータへのアクセスとデータ処理のための最優先要件は開発におけるセキュリティであることに気付きました。そこで、アジャイル SDLC (ソフトウェア開発ライフサイクル) の基盤として ASVS を使用することを選択しました。

ASVS を使用して、ログイン機能の最適な実装方法などの機能的なセキュリティの問題について、ユーザーストーリーとユースケースを生成します。スタートアップ企業は、通常とは異なる方法で ASVS を使用します。つまり、ASVS を精査し、現在のスプリントに適合する要件を拾い出します。そして、機能的要件はスプリントのバックログに直接追加し、非機能的要件は制約として既存のユースケースに追加します。たとえば、パスワードポリシーとブルートフォースの検出と防止機構としての効果を倍増させる Web サービスレギュレータと合わせて、TOTP 2 要素認証の追加を選択します。以降のスプリントでは、その他の要件を "JIT" (just in time = 土壇場で追加する)、"YAGNT" (You ain't gonna need it = 必要でないなら追加しない) 方式で選択します。

開発者は ASVS をピアレビュー用チェックリストとして使用し、安全でないコードがチェックインされないようにします。また、ASVS を使用して、過去に遡って新機能をチェックインした開発者を綿密に調査して適切な ASVS 要件を考慮していることを確認し、将来のスプリントで改善または緩和できるものはないかチェックします。

最後に、開発者は、ASVS を自動検査のセキュアな単体 / 統合テストの一部として使用し、ユーステストケース、悪用テストケース、ファジーテストケースをそれぞれテストします。この目的は、マイルストーンのビルドを製品としてリリースする場合に、ウォーターフォール手法における "開発の最終段階のペネトレーションテスト" で生じるリスクを軽減することにあります。各スプリント後に新規のビルドが昇格される可能性があるため、単一の保証活動に頼るのでは不十分です。テスト体制を自動化することによって、熟練したペネトレーションテスト担当者をもってしても数週間もテストしなければ見つけられないような重大な問題もなくなります。

ソフトウェアが検査レベルを満たすことの評価

ASVS 認定と信頼マークに関する OWASP の立ち位置

OWASP は、ベンダー中立の非営利組織であり、ベンダー、認証者、ソフトウェアの認定は一切行っていません。

そのような保証の表明、信頼マーク、認定は、いずれも OWASP によって正式に検査、登録、または認定されたものではありません。認定に期待を置く組織は、ASVS 認定を主張するあらゆる第三者または信頼マークについて、その信頼性を慎重に検討する必要があります。

ただし、OWASP の正式な認定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。

組織認定の指針

Application Security Verification Standard は、アプリケーションの公開検査にも活用することができます。公開検査においては、設計担当者や開発者、プロジェクトドキュメント、ソースコード、特に L2 および L3 検査用に使用できるテストシステムへの認証アクセス (各役割について最低 1 つのアカウントへのアクセス) など、アプリケーションの重要リソースに対して制限なく自由にアクセスすることができます。

歴史的に、ペネトレーションテストとセキュアコードのレビューは例外に基づいて問題を取り扱ってきました。つまり、テストに不合格であった問題のみが最終報告書に取り込まれるのは、エラーがある問題だけです。しかし、認定機関は、報告書には必ず、検査の範囲 (特に、SSO 認証などの重要な構成要素が範囲外である場合) と、合格したテストと不合格のテストなどの検査結果の要約、および不合格のテストに対する解決策の明確な指示を含める必要があります。

詳細な書類、スクリーンショットやムービー、問題を確実かつ反復的に再現するスクリプト、テストの電子的記録 (たとえば、受信プロキシのログや、クリーンアップリストなどの関連メモ) を保存しておくことは、業界の標準的な慣行と見なされており、懐疑的な開発者に対する検査結果の証拠として非常に役立つことがあります。単にツールを実行してエラーをするだけでは不十分です。それだけでは、認定対象のレベルのすべての問題を完全にテ

ストしたことを裏付ける十分な証明には (まったく) なりません。論争になったときには、検査対象の要件のすべてをそれぞれテストしたことを確実に実証する十分な証明が必要になります。

自動ペネトレーションテストツールの役割

自動のペネトレーションテストツールは、できるだけ広いカバレッジを提供し、かつ多くの異なる形式の悪性入力をテストできることが推奨されます。

自動ペネトレーションテストツールのみを使用して、ASVS の検査のすべてを完全に実行することはできません。L1 の要件については、その大半を自動テストで検査できますが、全体的には、ほとんどの要件が自動ペネトレーションテストにはなじみません。

ただし、自動テストと手動テストとの間の線引きは、アプリケーションセキュリティ産業の成熟化が進むにつれて、不明確になってきています。自動ツールは多くの場合、技術担当者によって手動調整され、また手動テストの担当者がさまざまな自動ツールを活用することもよくあります。

ペネトレーションテストの役割

手動によるペネトレーションテストを実行して、ソースコードにアクセスすることなく、L1 のすべての問題を検査することは可能ですが、これは主流の手法ではありません。L2 では、少なくとも、開発者、ドキュメント、コードへの何らかのアクセスと、システムへの認証アクセスが必要になります。レベル 3 では、ペネトレーションテストですべてに対応するのは不可能です。なぜなら、レベル 3 の追加項目のほとんどは、システムの構成のレビュー、悪性コードのレビュー、脅威モデリングなどの非ペネトレーションテストの成果物を必要とするためです。

セキュリティアーキテクチャの詳細な指針として

Application Security Verification Standard の一般的な使用法の 1 つは、セキュリティ設計担当者用のリソースとして使用することです。2 つの主要セキュリティアーキテクチャフレームワーク、SABSA と TOGAF には、アプリケーションセキュリティアーキテクチャのレビューを実施する際に必要な多くの情報が不足しています。ASVS を採用することで、セキュリティ設計担当者は、データ保護パターンや入力検証戦略などの一般的な問題をより適切に管理できるようになるため、これらのフレームワークのギャップを埋めることができます。

既製のセキュアコーディングチェックリストに代わるものとして

多くの組織は、ASVS の 3 つのレベルのいずれかを選択する、あるいは、ASVS から分岐し、各アプリケーションリスクレベルの要件を分野固有の方法で変更する形で、ASVS を活用することができます。我々はこのタイプの分岐を奨励していますが、前提条件として、追跡可能性が維持されていることが必要です。それにより、たとえばアプリケーションが要件 4.1 に合格している場合、分岐が進んでいっても、分岐したコピーが ASVS 標準と同じ要件を満たすことがわかります。

自動単体および統合テストのガイドとして

ASVS は、アーキテクチャと悪性コードに関する要件を除いて、テスト可能性がきわめて高いものとなっています。特定の関連するファジングのケースや悪用ケースに対するテストを行う単体テストと統合テストを構築することによって、アプリケーションはすべてのビルドでほぼ自己検査可能になります。たとえば、ログインコントローラー用のテストスイートの追加テストを作成し、一般的なユーザー名、アカウントの列挙、ブルートフォース攻撃、LDAP/SQL インジェクション、および XSS に対してユーザー名パラメーターをテストすることができます。同様に、パスワードパラメーターに関するテストには、一般的なパスワード、パスワードの長さ、ナルバイトインジェクション、パラメーターの削除、XSS、アカウントの列挙などを組み込みます。

セキュア開発のトレーニングとして

ASVS はセキュアソフトウェアの特性の定義にも使用できます。"セキュアコーディング" コースの多くは、コーディングのヒントをほんの少し付け足した、単なる倫理的ハッキングコースにすぎません。これでは開発者の助けにはなりません。そこで、ASVS をセキュアな開発のコースで使用し、やってはいけないトップ 10 項目ではなく、ASVS に含まれる事前対処的な管理策に重点をおくことをお勧めします。

ASVS を使用する OWASP プロジェクト

Security Knowledge Framework

https://www.owasp.org/index.php/OWASP_Security_Knowledge_Framework

セキュアコーディングに関する開発者のトレーニング。SKF は、OWASP Application Security Verification Standard を使用した、チームメンバーに対してセキュアコーディングのトレーニングを行うための、オープンソースの Python-Flask Web アプリケーションです。

OWASP Zed Attack Proxy

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

OWASP Zed Attack Proxy (ZAP) は、使い勝手の良い Web アプリケーション内の脆弱性を検出するための統合ペネトレーションテストツールです。ZAP は、セキュリティ上の経験に関して幅広いユーザーが使用することを想定して設計されているため、ペネトレーションテストを初めて行う開発者や機能テスト担当者に最適なツールとなっています。ZAP は、自動スキャナーに加えて、脆弱性を手動で検出できる一連のツールを備えています。

OWASP Cornucopia

https://www.owasp.org/index.php/OWASP_Cornucopia

OWASP Cornucopia は、ソフトウェア開発チームがアジャイル開発、従来の開発、形式的開発プロセスにおいてセキュリティ要件の決定を支援する、カードゲームです。ゲームは言語、プラットフォーム、テクノロジには依存しません。Cornucopia スイートは、OWASP Secure Coding Practices - Quick Reference Guide (SCP) の構成に基づいて選択されましたが、OWASP Application Security Verification Standard、OWASP Testing Guide、および David Rook の Principles of Secure Development のセクションの内容も考慮されています。

詳細な検査要件

V1. アーキテクチャ、設計、脅威モデリング

V2. 認証

V3. セッション管理

V4. アクセス制御

V5. 悪性入力の処理

V7. 保存データの暗号化

V8. エラー処理とログの保存

V9. データの保護

V10. 通信

V11. HTTP に関するセキュリティ設定

V13. 悪性活動の適切な処理

V15. ビジネスロジック

V16. ファイルとリソース

V17. モバイル

V18. Web サービス (3.0 で新たに追加)

V19. 構成 (3.0 で新たに追加)

V1: アーキテクチャ、設計、脅威モデリング

管理目標

検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- レベル 1: アプリケーションの構成要素が特定され、各構成要素に存在理由があること
- レベル 2: アーキテクチャが定義されていて、コードがアーキテクチャに準拠していること
- レベル 3: アーキテクチャと設計が存在し、使用され、機能していること

注: このセクションはバージョン 3.0 で再導入されましたが、基本的には、バージョン 1.0 の ASVS におけるアーキテクチャ上の管理と同じです。

要件

番号	説明	1	2	3	導入
1.1	すべてのアプリケーションのすべての構成要素を把握し、それらが必要である。	✓	✓	✓	1.0
1.2	ライブラリ、モジュール、外部システムなど、アプリケーションに内包されてはいないがアプリケーションの動作に必要となる構成要素をすべて把握している。		✓	✓	1.0
1.3	アプリケーションの高次のアーキテクチャが定義されている。		✓	✓	1.0
1.4	アプリケーションのすべての構成要素が業務上の機能、セキュリティ上の機能について定義されている。			✓	1.0
1.5	アプリケーションに内包されていないがその動作に必要となるすべての構成要素が、業務上の機能、セキュリティ上の機能について定義されている。			✓	1.0
1.6	対象となるアプリケーションの脅威モデルが作成されており、STRIDE、つまり、なりすまし (Spoofing)、改ざん (Tampering)、否認 (Repudiation)、情報漏えい (Information Disclosure)、権限昇格 (Elevation of privilege) に関連するリスクが網羅されている。			✓	1.0
1.7	すべてのセキュリティ管理 (外部のセキュリティサービスを呼び出すライブラリを含む) が集中実装されている。			✓	1.0

1.8	ネットワークセグメント、ファイアウォールルール、クラウドベースのセキュリティグループなどの、定義されたセキュリティ管理を通じて、構成要素が相互に切り離されている。	✓	✓	3.0
1.9	アプリケーションでは、データレイヤー、コントローラーレイヤー、表示レイヤーが明確に切り離されており、信頼できるシステム上でセキュリティについての決定を適用できる。	✓	✓	3.0
1.10	センシティブなビジネスロジック、秘密鍵、その他の機密情報がクライアント側のコードに含まれていない。	✓	✓	3.0

参照先

詳細については、以下を参照してください。

- Threat Modeling Cheat Sheet: https://www.owasp.org/index.php/Application_Security_Architecture_Cheat_Sheet
- Attack Surface Analysis Cheat Sheet: https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet

V2: 認証に関する検査要件

管理目標

認証とは、対象が本物 (または本人) であること、つまり、対象についてなされた主張が真実であることを立証または確認する行為です。検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- 通信の送信元のデジタル ID が検査できること
- 認可された者だけが認証可能であり、資格情報はセキュアな方法で転送されること

要件

番号	説明	1	2	3	導入
2.1	公開を意図しているものを除き、意図していないすべてのページとリソースがデフォルトで認証を必要とする (完全仲介の原則)。	✓	✓	✓	1.0
2.2	すべてのパスワードフィールドについてはユーザーのパスワードがそのまま表示されない設定になっている。	✓	✓	✓	1.0
2.4	すべての認証の管理がサーバー側で行われる。	✓	✓	✓	1.0
2.6	攻撃者がログインできないように、認証失敗の場合の安全対策を施している。	✓	✓	✓	1.0
2.7	パスワード入力フィールドで、パスフレーズの使用を許可または推奨し、長いパスフレーズやきわめて複雑なパスワードの入力を拒否しない。	✓	✓	✓	3.0
2.8	アカウントへのアクセス回復に使用できるすべてのアカウント ID 認証機能 (プロファイル更新機能、パスワードを忘れた場合の機能、トークンが無効になった / 紛失した場合の機能、ヘルプデスク、IVR など) が、一義的な認証メカニズムとしての機能を備えており攻撃に耐えうる。	✓	✓	✓	2.0
2.9	パスワード変更機能には、古いパスワード、新しいパスワード、パスワードの確認が含まれている。	✓	✓	✓	1.0
2.12	すべての疑わしい認証関連の処理が記録されている。これには、リクエスト、およびセキュリティ調査に必要な関連メタデータが含まれていなければならない。		✓	✓	2.0

2.13	アカウントパスワードで十分な強度の暗号化ルーチンを活用しており、この暗号化ルーチンがブルートフォース攻撃に耐えられる。		✓	✓	3.0
2.16	資格情報は暗号化された適切なリンクを使用して転送され、ユーザーに資格情報の入力を求めるすべてのページや機能は暗号化されたリンクを使用して転送されている。	✓	✓	✓	3.0
2.17	パスワードを忘れた場合の機能や他の回復パスにより現在のパスワードが露呈することはなく、新しいパスワードがクリアテキストでユーザーに送信されることもない。	✓	✓	✓	2.0
2.18	ログイン機能、パスワード再設定機能、またはアカウントを忘れた場合の機能を使用して情報の列挙はできない。	✓	✓	✓	2.0
2.19	アプリケーションのフレームワーク、またはアプリケーションが使用するあらゆる構成要素でデフォルトパスワード ("admin/password" など) が使用されていない。	✓	✓	✓	2.0
2.20	ブルートフォース攻撃や DoS 攻撃などの一般的な認証攻撃に対して、自動攻撃を防止するためのリクエスト制限機能が存在している。	✓	✓	✓	3.0
2.21	外部サービスにアクセスする認証証明は暗号化されて、安全対策の施された環境で保管されている。		✓	✓	2.0
2.22	パスワードを忘れた場合等の復帰パスにおいて、ソフトトークン、モバイルプッシュ、またはオフラインの復帰メカニズムが使用されている。	✓	✓	✓	3.0
2.23	アカウントのロックアウトが、ソフトロックステータスとハードロックステータスに分割されており、これらは相互に排他的でない。ブルートフォース攻撃によりアカウントが一時的にロックアウトされた場合、ハードロックステータスがリセットされない。		✓	✓	3.0
2.24	知識に基づく質問 ("秘密の質問" とも言う) が求められる場合、その質問はアプリケーションを保護できる強力なものである。	✓	✓	✓	2.0
2.25	過去に使用されたパスワードの再使用禁止をシステムに設定できる。		✓	✓	2.0
2.26	アプリケーション固有の機密の操作がそのアプリケーションのリスクプロファイルに基づいて許可される前に、再認証、ステップアップ認証または適応認証、2 要素認証、あるいはトランザクション署名が要求される。		✓	✓	2.0
2.27	ありがちなパスワードや弱いパスフレーズの使用を禁止する対策が存在している。	✓	✓	✓	3.0
2.28	成功、失敗にかかわらず、すべての認証処理が同じ平均応答時間で応答される。			✓	3.0
2.29	シークレット、API 鍵、パスワードはソースコード内またはオンラインソースコードリポジトリ内に含まれていない。			✓	3.0

2.30	アプリケーションでユーザーの認証を許可する場合、セキュアな実証済み認証メカニズムを使用する。	✓	✓	✓	3.0
2.31	アプリケーションでユーザーの認証を許可する場合、2 要素認証または他の強力な認証、あるいはユーザー名 + パスワードの漏えいを防ぐ同様のスキームを使用して認証できる。		✓	✓	3.0
2.32	信頼できない人が管理インターフェイスにアクセスできない。	✓	✓	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0: Testing for Authentication: https://www.owasp.org/index.php/Testing_for_authentication
- Password Storage Cheat Sheet: https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet
- Forgot Password Cheat Sheet: https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet
- Choosing and Using Security Questions Cheet Sheet: https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet (See LoA)

V3: セッション管理に関する検査要件

管理目標

Web ベースアプリケーションの中核となる構成要素の 1 つは、アプリケーションと対話するユーザーの状態を管理 / 維持するためのメカニズムです。これはセッション管理と呼ばれ、ユーザーと Web ベースアプリケーション間のステートフルな対話を管理する管理策として定義できます。

検査対象のアプリケーションが次の高次のセッション管理要件を満たすことを確認します。

- セッションは個々のユーザー固有のものであり、推測や共有はできないこと
- セッションは、不要になると無効化され、非アクティブの状態が続くとタイムアウトするこ。

要件

番号	説明	1	2	3	導入
3.1	独自実装のセッションマネージャーは存在していない、または独自実装のセッションマネージャーは一般的なあらゆるセッション管理攻撃に耐えられる。	✓	✓	✓	1.0
3.2	ログアウト時にセッションを無効化している。	✓	✓	✓	1.0
3.3	ある一定の期間の非アクティブ状態でセッションがタイムアウトする。	✓	✓	✓	1.0
3.4	管理者が設定した時間になると、ユーザーがログイン状態であってもセッションタイムアウトとなる機能 (絶対タイムアウト機能) を備えている。			✓	1.0
3.5	認証を必要とするすべてのページに、わかりやすく簡単なログアウト機能がある。	✓	✓	✓	1.0
3.6	URL、エラーメッセージ、ログを通じてセッション ID が公開されない。セッションクッキーによる URL 書換えをサポートしない。	✓	✓	✓	1.0
3.7	認証、再認証が成功するたびに新たなセッションとセッション ID が生成される。	✓	✓	✓	1.0
3.10	アプリケーションフレームワークで生成したセッション ID のみを有効と認識する。		✓	✓	1.0
3.11	セッション ID が十分な長さを持ち、ランダムで、かつ正しいアクティブセッションベース全体で一意である。	✓	✓	✓	1.0

3.12	Cookie に保存されているセッション ID はアプリケーションのみに限定されたパスを適切に設定し、認証セッショントークンはさらに "HttpOnly" 属性と "secure" 属性を設定する。	✓	✓	✓	3.0
3.16	アプリケーションでアクティブな同時セッション数を制限している。	✓	✓	✓	3.0
3.17	各ユーザーのアカウントプロファイルまたはこれに類似するものにアクティブセッションリストが表示される。ユーザーは任意のアクティブセッションを終了できる。	✓	✓	✓	3.0
3.18	パスワード変更プロセスが正常に実行された後に、他のすべてのアクティブセッションを終了する選択肢がユーザーに表示される。	✓	✓	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0:Session Management Testing: https://www.owasp.org/index.php/Testing_for_Session_Management
- OWASP Session Management Cheat Sheet: https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

V4: アクセス制御に関する検査要件

管理目標

認可とは、リソースへのアクセスを、当該リソースの使用を許可されているユーザーのみに制限する概念です。検査対象のアプリケーションが次の高次の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ーが有効な資格情報を保持していること
- ユーザーが、正しく定義されたロールと権限のセットに関連付けられていること
- ロールとアクセス許可のメタデータが、再生または改ざんされないよう保護されていること

要件

番号	説明	1	2	3	導入
4.1	最小権限の原則が存在している。ユーザーは認可されている場合のみ、機能、データファイル、URL、コントローラー、サービス、他のリソースにアクセスできる。これは、なりすましおよび権限昇格に対する防御となる。	✓	✓	✓	1.0
4.4	機密レコードへのアクセスは保護されており、各ユーザーがアクセスできるのは認可されたオブジェクトまたはデータのみである (たとえば、パラメーターの改ざんにより別のユーザーのアカウントを表示 / 変更するのを防ぐ)。	✓	✓	✓	1.0
4.5	ディレクトリー覧表示は、意図的な許可がない限り、不可能な状態となっている。また、ファイルまたはディレクトリメタデータ (Thumbs.db、.DS_Store、.git、.svn フォルダーなど) の検出や開示は不可能である。	✓	✓	✓	1.0
4.8	アクセス制御に失敗した場合に安全な処理が施されている。	✓	✓	✓	1.0
4.9	プレゼンテーション層と同じアクセス制御がサーバー側で実行される。	✓	✓	✓	1.0
4.10	アクセス制御で利用している全ユーザー属性、データ属性、ポリシー情報をエンドユーザーは特別の許可なく変更できない。		✓	✓	1.0
4.11	保護されているリソースへのアクセスを保護のための集中管理メカニズムを備えている。これには外部認可サービスを利用するライブラリも含まれる。			✓	1.0
4.12	アクセスの制御および制御失敗が記録されている。		✓	✓	2.0
4.13	アプリケーションまたはフレームワークが、強力なランダムのアンチ CSRF トーク	✓	✓	✓	2.0

	ンを使用しているか、または別のトランザクション保護メカニズムを備えている。				
4.14	システムが、セキュアな機能、リソース、データへの集約的または連続的なアクセスを防止できる。リソースガバナーを使用して、1 時間あたりの編集数を制限する、個別ユーザーによるデータベース全体のスクレイピングを防止する、などの対策を取る。		✓	✓	2.0
4.15	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)、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の不正行為に基づいて不正行為対策の制御を行う。		✓	✓	3.0
4.16	アプリケーションで状況依存の認可を適切に行い、パラメーターの改ざんによる、認可されていない操作の実行が不可能である。	✓	✓	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0: Authorization: https://www.owasp.org/index.php/Testing_for_Authorization
- OWASP Cheat Sheet:Access Control: https://www.owasp.org/index.php/Access_Control_Cheat_Sheet

V5: 悪性入力の処理に関する検査要件

管理目標

Web アプリケーションセキュリティの最も一般的な脆弱性は、クライアントまたは環境から受信した入力が検証されないまま使用されることにあります。この脆弱性は、クロスサイトスクリプティング、SQL インジェクション、インタープリターインジェクション、ロケール / Unicode 攻撃、ファイルシステム攻撃、バッファーオーバーフローなど、Web アプリケーションの主な脆弱性のほとんどすべてにつながります。

検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- すべての入力が正しく、本来の目的に適合していることが確認されている
- 外部のエンティティまたはクライアントからのデータは、信頼できないものとして扱われること

要件

番号	説明	1	2	3	導入
5.1	ランタイム環境がバッファーオーバーフローの影響を受けにくい。またはバッファーオーバーフローの対策を持っている。	✓	✓	✓	1.0
5.3	サーバー側で入力検証エラーが発生した場合、リクエストを拒否し、ログに記録する。	✓	✓	✓	1.0
5.5	入力検証ルーチンはサーバー側で実施する。	✓	✓	✓	1.0
5.6	アプリケーションに許可される各データ型に対して、単一の入力検証機構が使用されている。			✓	1.0
5.10	すべての SQL クエリ、HQL、OSQL、NOSQL、ストアドプロシージャ、ストアドプロシージャの呼び出しが、プリペアドステートメントまたはクエリのパラメーター化の使用によって保護されており、SQL インジェクションの影響を受けない。	✓	✓	✓	2.0
5.11	アプリケーションは 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制御によって LDAP インジェクションが防止される。	✓	✓	✓	2.0
5.12	アプリケーションは OS コマンド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制御によって OS コマンドインジェクションが防止される。	✓	✓	✓	2.0

5.13	ファイルのパスであるコンテンツが使用されたときに、アプリケーションが、リモートファイルインクルード (RFI) またはローカルファイルインクルード (LFI) の影響を受けない。	✓	✓	✓	3.0
5.14	アプリケーションは、XPath クエリの改ざん、XML 外部エンティティ攻撃、XML インジェクション攻撃などの一般的な XML 攻撃の影響を受けない。	✓	✓	✓	2.0
5.15	HTML または他の Web クライアントコードに存在するすべての文字列変数が、コンテキストに応じて適切に手動でエンコードされているか、またはコンテキストに応じた自動エンコードのテンプレートを活用しており、アプリケーションは、反射型、格納型、および DOM クロスサイトスクリプティング (XSS) 攻撃の影響を受けない。	✓	✓	✓	2.0
5.16	アプリケーションフレームワークが、受信リクエストからモデルへの自動一括パラメーター割り当て (自動変数バインドとも呼ばれる) を許可している場合、"*accountBalance*"、"*role*"、"*password*" などのセキュリティ上の機密フィールドが悪意のある自動バインドから保護されている。		✓	✓	2.0
5.17	アプリケーションが HTTP パラメーター汚染攻撃に対する防御を備えている。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ー (GET、POST、Cookie、ヘッダー、環境など) のソースを区別しない場合は、特にこの防御が必要。		✓	✓	2.0
5.18	サーバー側の検証に加えて、クライアント側の検証が第 2 の防衛線として使用されている。		✓	✓	3.0
5.19	HTML フォームフィールドだけでなく、REST 呼び出し、クエリパラメーター、HTTP ヘッダー、Cookie、バッチファイル、RSS フィードなどの入力ソースのすべての入力データが、ポジティブ検証 (ホワイトリスト) を使用し、その後グレーリスト (既知の不適切文字列の排除) などの副次的な検証形式を使用して検証されるか、または不適切な入力を拒否する (ブラックリスト) ことによって検証される。		✓	✓	3.0
5.20	構造化データが、厳密に型指定されており、使用可能な文字、長さ、パターンなどの定義されたスキーマに基づいて検証される (たとえば、クレジットカード番号や電話番号などの検証、または地区名と郵便番号などの 2 つの関連するフィールドのデータが妥当であることの検証)。		✓	✓	3.0
5.21	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の一般的な安全対策が適用されており、特定のコンテキストで有害な可能性がある文字 (たとえば、"ねこ" や "O'Hara" など、Unicode やアポストロフィを含む普通の名前) がエスケープ処理されている。		✓	✓	3.0
5.22	WYSIWYG エディターまたは同様のものからの信頼できない HTML が HTML サニタイザーによって適切に無害化され、入力検証タスクまたはエンコードタスクに従って適切に処理されている。	✓	✓	✓	3.0
5.23	自動エスケープテンプレート機構について、UI のエスケープが無効な場合は、代		✓	✓	3.0

	わりに HTML の無害化が有効になっている。			
5.24	1 つの DOM コンテキストから別の DOM コンテキストへ転送されるデータで安全な JavaScript メソッドが使用される (.innerText や .val の使用など)。	✓	✓	3.0
5.25	ブラウザーでの JSON の解析時に、クライアントで JSON.parse を使用して JSON が解析される。クライアントでの JSON の解析に eval() を使用しない。	✓	✓	3.0
5.26	セッションの終了後、ブラウザー DOM などの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。	✓	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0: Input Validation Testing: https://www.owasp.org/index.php/Testing_for_Input_Validation
- OWASP Cheat Sheet:Input Validation: https://www.owasp.org/index.php/Input_Validation_Cheat_Sheet
- OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution: https://www.owasp.org/index.php/Testing_for_HTTP_Parameter_pollution_%28OTG-INPVAL-004%29
- OWASP LDAP Injection Cheat Sheet: https://www.owasp.org/index.php/LDAP_Injection_Prevention_Cheat_Sheet
- OWASP Testing Guide 4.0: Client Side Testing: https://www.owasp.org/index.php/Client_Side_Testing
- OWASP Cross Site Scripting Prevention Cheat Sheet: https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet
- OWASP Java Encoding Project: https://www.owasp.org/index.php/OWASP_Java_Encoder_Project

自動エスケープの詳細については、以下を参照してください。

- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems: http://googleonlinesecurity.blogspot.com/2009/03/reducing-xss-by-way-of-automatic.html
- AngularJS Strict Contextual Escaping: https://docs.angularjs.org/api/ng/service/$sce

V6: 出力のエンコード / エスケープ

このセクションは、Application Security Verification Standard 2.0 の V5 に組み込まれました。ASVS 要件 5.16 では、クロスサイトスクリプティングの防止に役立つ、コンテキストに応じた出力のエンコードに対処しています。

V7: 保存データの暗号化に関する検査要件

管理目標

検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- すべての暗号化モジュールが失敗した場合の安全対策を施しており、エラーが適切に処理されること
- ランダム性が必要な場合は適切な乱数生成器が使用されること
- 鍵へのアクセスが安全な方法で管理されていること

要件

番号	説明	1	2	3	導入
7.2	すべての暗号化モジュールは処理失敗の場合には安全に対処できるように対策が立てられており、オラクルパディングが不可能な方法でエラーが処理される。	✓	✓	✓	1.0
7.6	すべての乱数、ランダムなファイル名や GUID、ランダムな記号文字列は、攻撃者から推測されないようにする場合、暗号化モジュールが許可した乱数生成器を用いて乱数を生成する。		✓	✓	1.0
7.7	アプリケーションが使用する暗号化アルゴリズムは FIPS 140-2 または同等の標準で検証されたものを使用する。	✓	✓	✓	1.0
7.8	暗号化モジュールがセキュリティポリシーに則って活用されている。			✓	1.0
7.9	暗号鍵の管理運営について明確なポリシーが規定されており (生成、提供、廃棄、無効など)、正しく実施されている。		✓	✓	1.0
7.11	暗号化サービスのすべてのユーザーは鍵マテリアルに直接アクセスできない。マスターシークレットを含む暗号化プロセスを分離し、ハードウェア鍵コンテナー (HSM) の使用を検討する。			✓	3.0
7.12	*個人識別情報*は、保存時には暗号化されており、通信時には保護されたチャネルを介してやりとりされる。		✓	✓	3.0
7.13	可能な場合、鍵およびシークレットは破棄時にゼロ設定する。		✓	✓	3.0
7.14	すべての鍵とパスワードが置換可能であり、インストール時に生成または置換される。		✓	✓	3.0

7.15	アプリケーションの負荷が高いときも、乱数は適切なエントロピーで生成される、またはアプリケーションが正常な形でパフォーマンスを低下させる。	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0: Testing for weak Cryptography: https://www.owasp.org/index.php/Testing_for_weak_Cryptography
- OWASP Cheat Sheet: Cryptographic Storage: https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

V8: エラー処理とログの保存に関する検査要件

管理目標

エラー処理とログの保存の主な目標は、ユーザー、管理者、およびインシデント対応チームを適切な形で支援することです。目標は大量のログ生成ではなく、不要なノイズよりシグナルのほうが多い、高品質のログの生成です。

高品質のログには、多くの場合、機密のデータが含まれるため、各データプライバシー法または条令に従って保護する必要があります。これには次のことが含まれます。

- 必要な場合を除いて、機密情報の収集やログの保存は行わない
- ログに保存された情報はすべてデータ分類に基づいて、セキュアな方法で処理し、保護する
- ログは永続的なものとはせず、絶対的有効期間をできるだけ短く設定する

ログにプライベートまたは機密データ (この定義は国によって異なる) が含まれる場合、ログはアプリケーションが保持する最高機密情報となるため、攻撃者にとってきわめて魅力的なものとなります。

要件

番号	説明	1	2	3	導入
8.1	攻撃者に利用される可能性がある機密データ (セッション ID、ソフトウェア / フレームワークバージョン、個人情報など) はエラーメッセージまたはスタックトレースに表示しない。	✓	✓	✓	1.0
8.2	セキュリティに関連したエラー処理に関するロジックへのアクセスがデフォルトで許可されていない。		✓	✓	1.0
8.3	セキュリティに関するログの保存では、セキュリティに関わるイベントであれば、成功、失敗両方のログを保存できるように設定する。		✓	✓	1.0
8.4	各ログイベントには、イベント発生時のタイムラインの詳細な調査のために必要な情報が含まれている。		✓	✓	1.0
8.5	信頼できないデータを含むすべてのイベントは、ログ閲覧ソフトによってコードとして実行されない。			✓	1.0

8.6	セキュリティログが認可されていないアクセスや修正から保護されている。	✓	✓	1.0
8.7	各プライバシー法や規制で定義されている機密データ、リスク評価で定義されている組織の機密データ、あるいは攻撃者に利用される可能性がある機密の認証データ (ユーザーのセッション ID、パスワード、ハッシュ、API トークンなど) はログとして保存しない。	✓	✓	3.0
8.8	すべての印刷不可能なシンボルやフィールド区切り記号が、ログエントリ内で適切にエンコードされ、ログインジェクションが防止される。		✓	2.0
8.9	ログエントリ内で、信頼できるソースからのログフィールドと信頼できないソースからのログフィールドが区別できる。		✓	2.0
8.10	監査ログまたは同様のものによって、主要トランザクションの否認不可が可能である。	✓	✓	3.0
8.11	セキュリティログの認可されていない変更を防止するための、何らかの完全性チェックまたは制御が備わっている。		✓	3.0
8.12	ログはアプリケーションの実行場所とは異なるパーティションに保存されており、また適切なログローテーションが適用されている。		✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0 のコンテンツ: Testing for Error Handling: https://www.owasp.org/index.php/Testing_for_Error_Handling

V9: データの保護に関する検査要件

管理目標

適切なデータ保護の主な要素として、次の 3 つがあります。CIA、つまり機密性 (Confidentiality)、完全性 (Integrity)、可用性 (Availability) です。この標準は、データ保護が、サーバーなど強力かつ十分な保護を備えた信頼できるシステム上で実施されることを前提とします。アプリケーションは「ユーザーデバイスはどれも完全には信頼できない」ことを前提にしている必要があります。アプリケーションが共有のコンピューター、電話、タブレットなどの安全でないデバイスに機密データを送信または保存する場合、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、変更、開示が容易にはできないようにする必要があります。

検査対象のアプリケーションが次の高次のデータ保護要件を満たすことを確認します。

- 機密性: 送信中と保存時の両方で、認可されていない監視や開示からデータを保護すること。
- 完全性: 認可されていない攻撃者による悪意のある作成、変更、削除からデータを保護すること。
- 可用性: 必要に応じて、認可されたユーザーに対してデータを利用可能にすること。

要件

番号	説明	1	2	3	導入
9.1	機密情報を扱うすべてのフォームがクライアント側でのキャッシュ (オートコンプリート機能を含む) を許可していない。	✓	✓	✓	1.0
9.2	アプリケーションによって処理される機密データのリストが特定されており、関連するデータ保護条令に基づいてこれら機密データへのアクセスの制御、暗号化、実施に関するポリシーが明文化されている。			✓	1.0
9.3	すべての機密データは HTTP メッセージ本文またはヘッダーでサーバーに送信する (URL パラメーターを使用した機密データ送信は行わない)。	✓	✓	✓	1.0
9.4	アプリケーションのリスクに基づいて、次のような適切なアンチキャッシュヘッダーがアプリケーションによって設定される。 Expires: Tue, 03 Jul 2001 06:00:00 GMT Last-Modified: {now} GMT Cache-Control: no-store, no-cache, must-revalidate, max-age=0	✓	✓	✓	1.0

	Cache-Control: post-check=0, pre-check=0 Pragma: no-cache				
9.5	サーバーに保存されている機密データのキャッシュまたは一時的なコピーは、認可されていないアクセスから保護され、認可されたユーザーが機密データにアクセスした後は無効化するか、または除去する。		✓	✓	1.0
9.6	保持ポリシーの終了時に各種機密データをアプリケーションから削除する方法が存在する。			✓	1.0
9.7	アプリケーションで、非表示フィールド、Ajax 変数、Cookie、ヘッダー値など、個別リクエスト内のパラメーター数が最小化されている。		✓	✓	2.0
9.8	アプリケーションが、スクリーンスクレイピングなど、データ収集のための異常な数のリクエストを検出し、アラートを生成する機能を備えている。			✓	2.0
9.9	クライアント側記憶域 (HTML5 ローカル記憶域、セッション記憶域、IndexedDB、通常の Cookie、Flash Cookie など) の保存データには機密データまたは PII は含まれていない。	✓	✓	✓	3.0
9.10	関連するデータ保護条令に基づいて機密データが収集される場合、またはアクセスのログ記録が必要とされる場合、機密データへのアクセスがログ記録される。		✓	✓	3.0
9.11	機密データは不要になった時点ですぐにメモリからサニタイズされ、フレームワーク / ライブラリ / オペレーティングシステムでサポートされている機能および技法に従って処理される。		✓	✓	3.0

参照先

詳細については、以下を参照してください。

- User Privacy Protection Cheat Sheet: https://www.owasp.org/index.php/User_Privacy_Protection_Cheat_Sheet

V10: 通信セキュリティに関する検査要件

管理目標

検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- 機密データ送信には TLS を使用すること。
- 常に強力なアルゴリズムと暗号を使用すること。

要件

番号	説明	1	2	3	導入
10.1	信頼できる CA から各トランスポート層セキュリティ (TLS) サーバー証明書までのパスを作成することができ、また各サーバー証明書が有効である。	✓	✓	✓	1.0
10.3	認証処理や、機密データ / 機密機能を処理する通信は、外部通信およびバックエンド方式による通信を含めてすべて TLS を使用しており、また安全性の低いプロトコルや非暗号化プロトコルにフォールバックしない。最も強力な代替策は推奨アルゴリズムである。	✓	✓	✓	3.0
10.4	バックエンド TLS 通信が失敗した場合は記録する。			✓	1.0
10.5	認証パスを構築し、トラストアンカーと失効情報を用いて、クライアント証明書を確認する。			✓	1.0
10.6	機密情報や機密機能を含む外部システムとの通信には認証を用いる。		✓	✓	1.0
10.8	承認された動作モードで機能するように構成されたアプリケーションがあり、そのアプリケーションが使用する標準 TSL が実装されている。			✓	1.0
10.10	TLS 証明書公開鍵のピン留めが、実稼働やバックアップの公開鍵で実装されている。詳細については、下記の参照先を参照。			✓	3.0
10.11	すべてのリクエストに、また Strict-Transport-Security: max-age=15724800; includeSubdomains などのすべてのサブドメインに対して、HTTP Strict Transport Security ヘッダーが含まれる。	✓	✓	✓	3.0
10.12	実稼働 Web サイト の URL が、Web ブラウザーベンダーが管理する、事前に読み込まれた Strict Transport Security ドメインリストに送信されている。下記の参			✓	3.0

	照先を参照。				
10.13	受動的な攻撃者によってトラフィックが記録されるリスクを軽減するために、Forward Secrecy 暗号が使用されている。	✓	✓	✓	3.0
10.14	Online Certificate Status Protocol (OCSP) の関連付けなど、証明書失効をチェックできる適切な機能が有効化され、設定されている。	✓	✓	✓	3.0
10.15	選択した認証機関のルート証明書や中間証明書を含むすべての証明書階層で、強力なアルゴリズム、暗号、プロトコルのみが使用されている。	✓	✓	✓	3.0
10.16	TLS の設定が、現在の推奨手法に合致している (一般的な構成、暗号、およびアルゴリズムの安全性は低下することがあるため)。	✓	✓	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP - TLS Cheat Sheet: https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- "承認されている TLS のモード" に関する注意: 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難しく、矛盾があり、また混乱を招く可能性があります。10.8 に準拠するためのより適切な方法は、https://wiki.mozilla.org/Security/Server_Side_TLS) などのガイドを確認し、既知の適切な構成を生成し (https://mozilla.github.io/server-side-tls/ssl-config-generator/)、sslyze などの既知の TLS 評価ツール、さまざまな脆弱性スキャナー、信頼できる TLS 評価サービスを使用して、必要なレベルのセキュリティを確保することです。一般に、このセクションへの非準拠のケースとして、旧式または安全性の低い暗号やアルゴリズムの使用、Perfect Forward Secrecy の欠如、旧式または安全性の低い SSL プロトコル、脆弱な推奨暗号などが見られます。
- 証明書のピン留め: 詳細については、https://tools.ietf.org/html/rfc7469 を確認してください。実稼働鍵とバックアップ鍵に証明書のピン留めを行う根拠は、ビジネス継続性にあります。https://noncombatant.org/2015/05/01/about-http-public-key-pinning/ を参照してください。
- HTTP Strict Transport Security の事前読み込み: https://www.chromium.org/hsts

V11: HTTP のセキュリティ設定に関する検査要件

管理目標

検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- アプリケーションサーバーがデフォルト構成から適切に強化されている。
- HTTP レスポンスの Content-Type ヘッダーに安全な文字セットが含まれている。

要件

番号	説明	1	2	3	導入
11.1	アプリケーションは、GET や POST などの、定義された必要な HTTP リクエストメソッドセットのみを受け付け、使用されていないメソッド (TRACE、PUT、DELETE など) は明示的にブロックする。	✓	✓	✓	1.0
11.2	すべての HTTP レスポンスに安全な文字セット (UTF-8 や ISO 8859-1 など) を指定したコンテンツタイプヘッダーを含んでいる。	✓	✓	✓	1.0
11.3	ベアラートークンなど、信頼できるプロキシまたは SSO デバイスによって追加された HTTP ヘッダーがアプリケーションによって認証される。		✓	✓	2.0
11.4	サードパーティの X-Frame 内でのコンテンツ表示を許可しないサイトについては Content Security Policy V2 (CSP) が使用されていることを確認する。		✓	✓	2.0
11.5	HTTP ヘッダー、または HTTP レスポンスのいかなる部分にも、システム構成要素の詳細なバージョン情報が公開されない。	✓	✓	✓	2.0
11.6	すべての API レスポンスに、X-Content-Type-Options: nosniff と Content-Disposition: attachment; filename="api.json" (または、そのコンテンツタイプの他の適切なファイル名) が含まれている。	✓	✓	✓	3.0
11.7	Content Security Policy V2 (CSP) が使用されており、この使用によって、インライン JavaScript が無効化されるか、あるいは CSP nonce またはハッシュに基づく、インライン JavaScript に対する完全性チェックが提供される。	✓	✓	✓	3.0
11.8	X-XSS-Protection: 1; mode=block ヘッダーが存在する。	✓	✓	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0: Testing for HTTP Verb Tampering: https://www.owasp.org/index.php/Testing_for_HTTP_Verb_Tampering_%28OTG-INPVAL-003%29

V12: セキュリティ設定に関する検査要件

このセクションは、Application Security Verification Standard 2.0 の V11 に組み込まれました。

V13: 悪性活動の適切な処理に関する検査要件

管理目標

検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- 検出された悪性活動が、アプリケーションの他の部分に影響しないように、セキュアな方法で適切に処理される。

要件

番号	説明	1	2	3	導入
13.1	すべての悪性活動が、サンドボックス化、コンテナ化、または隔離され、攻撃者が他のアプリケーションを攻撃することを阻止する。			✓	2.0
13.2	コードレビューにおいて、悪性コード、バックドア、イースターエッグ、およびロジック上の欠陥が検査される。			✓	3.0

V14: 内部セキュリティに関する検査要件

このセクションは、Application Security Verification Standard 2.0 の V13 に組み込まれました。

V15: ビジネスロジックに関する検査要件

管理目標

検査対象のアプリケーションが次の高次の要件を満たすことを確認します。

- ビジネスロジックフローが逐次的で、正しく順序付けられている。

要件

番号	説明	1	2	3	導入
15.1	ビジネスロジックフローがステップごとに逐次実行され、すべてのステップが人間が処理できる現実的な時間で処理され、誤った順序での処理やステップの省略、別のユーザーのステップの処理、送信されるのが早すぎるトランザクションの処理はいずれも行われない。		✓	✓	2.0
15.2	アプリケーションにビジネス上の制限が実装されており、ユーザー単位でそれが行われ、自動のあるいは異常な攻撃に対して、設定により警告をあげたり自動的に対応することができる。		✓	✓	2.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0: Business Logic Testing: https://www.owasp.org/index.php/Testing_for_business_logic
- OWASP Cheat Sheet: https://www.owasp.org/index.php/Business_Logic_Security_Cheat_Sheet

V16: ファイルとリソースに関する検査要件

管理目標

検査対象のアプリケーションが次の高次の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は、セキュアな方法で適切に処理されること。
- 信頼できない情報源から取得したデータは、webroot の外部に保存され、アクセス許可が制限されること。

要件

番号	説明	1	2	3	導入
16.1	URL のリダイレクトおよび転送で、ホワイトリストに含まれる宛先のみが許可される、または信頼できないコンテンツへのリダイレクト時には警告が表示される。	✓	✓	✓	2.0
16.2	アプリケーションに送信される信頼できないファイルのデータは、ファイル入出力コマンドで直接使用されない (特に、パストラバーサル、ローカルファイルインクルード、ファイルの MIME の種類、および OS コマンドインジェクションの脆弱性から守るため)。	✓	✓	✓	2.0
16.3	信頼できない情報源から取得したファイルが、想定された種類のものであることを検証するとともにウイルス対策ソフトで検査して、既知の悪性コンテンツがアップロードされることを防止する。	✓	✓	✓	2.0
16.4	信頼できないデータはインクルード、クラスローダー、またはリフレクション機能内で使用されず、リモート / ローカルファイルインクルード脆弱性に対する防御が行われる。	✓	✓	✓	2.0
16.5	信頼できないデータはクロスドメインリソース共有 (CORS) 内で使用されず、任意のリモートコンテンツに対する防御が行われる。	✓	✓	✓	2.0
16.6	信頼できない情報源から取得したファイルは webroot の外部に保存され、アクセス許可が制限されている。さらに、強力な検証が行われる。		✓	✓	3.0
16.7	Web またはアプリケーションサーバーが、デフォルトで当該 Web またはアプリケーションサーバーの外部にあるリモートリソース / システムへのアクセ		✓	✓	2.0

	スを拒否するように構成されている。				
16.8	信頼できない情報源から取得したデータをアプリケーションが実行しない。	✓	✓	✓	3.0
16.9	Flash、Active-X、Silverlight、NACL、クライアントサイド Java、W3C ブラウザー標準でネイティブにサポートされていないクライアントサイド技術を使用していない。	✓	✓	✓	2.0

参照先

詳細については、以下を参照してください。

- File Extension Handling for Sensitive Information: https://www.owasp.org/index.php/Unrestricted_File_Upload
- Unrestricted File Uploads: https://www.owasp.org/index.php/Unrestricted_File_Upload

V17: モバイルに関する検査要件

管理目標

- API や Web サービスといったサーバー側の管理策が、デバイス自体が備えるのと同レベルの管理策が存在すること。
- デバイス上に保存されるセンシティブな情報資産は、セキュアな方法で保存されること。
- デバイスから送信されるセンシティブなデータはすべて、トランスポート層セキュリティを考慮して送信されること。

注: バージョン 3.0 で廃止された要件はすべて、モバイルの要件が、この標準内の他の部分に含まれる要件と重複していることによります。これらの要件は重複しているため削除されました。各レベルでモバイルアプリケーション評価を完全に行うためには、この標準全体の他のすべてのアプリケーション要件についても考慮する必要があります。

要件

番号	説明	1	2	3	導入
17.1	UDID や IMEI 番号などデバイスに保存され、他のアプリケーションから取得可能な ID 値は、認証トークンとして使用されない。	✓	✓	✓	2.0
17.2	モバイルアプリは、デバイス上の暗号化されていない可能性がある共有リソース (SD カードや共有フォルダーなど) にセンシティブなデータを保存しない。	✓	✓	✓	2.0
17.3	センシティブなデータが保護されない状態でデバイス上に保存されていない (鍵チェーンなどの保護されたシステムエリア内を含む) 。	✓	✓	✓	2.0
17.4	秘密鍵、API トークン、パスワードがモバイルアプリケーション内で動的に生成される。		✓	✓	2.0
17.5	モバイルアプリがセンシティブな情報の漏えいを防止する (たとえば、現在のアプリケーションがバックグラウンドに切り替わるとき、またはコンソールに機密情報を書き込んでいるときに、アプリケーションのスクリーンショットが保存されるなど) 。		✓	✓	2.0
17.6	アプリケーションは必要な機能およびリソースに対する最小限のアクセス許可を要求している。		✓	✓	2.0

17.7	アプリケーションのセンシティブなコードがメモリに予測不可能な方法で配置される (ASLR など)。	✓	✓	✓	2.0
17.8	攻撃者によるモバイルアプリへのデバッガー (GDB など) のインジェクションを防止または遅延できるアンチデバッグ技法が導入されている。			✓	2.0
17.9	アプリは、センシティブなアクティビティ、インテント、コンテンツプロバイダーなどをエクスポートせず、同位置デバイス上の他のモバイルアプリでこれらが悪用されることがない。	✓	✓	✓	2.0
17.10	口座番号などのセンシティブな文字列に対して可変構造が使用されており、使用されないときには上書きされる (メモリ分析攻撃による被害の軽減)。			✓	2.0
17.11	アプリの公開されたアクティビティ、インテント、コンテンツプロバイダーなどにおいてすべての入力を検証する。	✓	✓	✓	2.0

参照先

詳細については、以下を参照してください。

- OWASP Mobile Security Project: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- IOS Developer Cheat Sheet: https://www.owasp.org/index.php/IOS_Developer_Cheat_Sheet

V18: Web サービスに関する検査要件

管理目標

RESTful または SOAP ベースの Web サービスを使用する検査対象のアプリケーションが、次の機能を備えていることを確認します。

- すべての Web サービスについて適切な認証、セッション管理、認可
- 低信頼レベルから高信頼レベルに遷移するすべてのパラメーターについての入力値検証
- API の使用を促進する、SOAP Web サービスレイヤーの基本的な相互運用性

要件

番号	説明	1	2	3	導入
18.1	クライアントとサーバー間で同じ符号化形式が使用されている。	✓	✓	✓	3.0
18.2	Web サービスアプリケーション内の管理機能にアクセスできるのは Web サービス管理者のみである。	✓	✓	✓	3.0
18.3	XML または JSON スキーマが存在し、入力の受付け前に検査される。	✓	✓	✓	3.0
18.4	すべての入力が適切なサイズに制限されている。	✓	✓	✓	3.0
18.5	SOAP ベースの Web サービスが、最低でも、Web サービス相互運用性 (WS-I) 基本プロファイルに準拠している。	✓	✓	✓	3.0
18.6	セッションベースの認証と認可が使用されている。詳細な指針については、セクション 2、3、4 を参照。静的な "API 鍵" および同類を使用しない。	✓	✓	✓	3.0
18.7	REST サービスが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。	✓	✓	✓	3.0
18.8	REST サービスにおいて受け取った Content-Type が想定通りのもの (application/xml や application/json など) であることを、明示的に確認している。		✓	✓	3.0
18.9	クライアントとサービス間でのデータのやりとりの信頼性を確保するためにメッセージペイロードを署名する。		✓	✓	3.0
18.10	セキュリティが低い代替のアクセスパスが存在しないことを確認する。		✓	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Web Services Security Cheat Sheet:
  https://www.owasp.org/index.php/Web_Service_Security_Cheat_Sheet
- OWASP Web Services Security Testing Cheat Sheet:
  https://www.owasp.org/index.php/Web_Service_Security_Testing_Cheat_Sheet

V19. 構成

管理目標

検査対象のアプリケーションが以下の要件を満たすことを確認します。

- 最新のライブラリおよびプラットフォームを備えていること。
- デフォルトの設定が安全であること。
- ユーザーによるデフォルト設定の変更によって、基礎システムのセキュリティの脆弱性 / 欠陥が不必要に露出、発生することがないよう、十分に機能強化されていること。

要件

番号	説明	1	2	3	導入
19.1	すべての構成要素が、適切なセキュリティ設定およびバージョンによって更新されている。不要な構成やフォルダー (サンプルアプリケーション、プラットフォームのドキュメント、デフォルトまたはサンプルユーザーなど) が削除されている。	✓	✓	✓	3.0
19.2	アプリケーションサーバーとデータベースサーバーの間など、構成要素間の通信が暗号化されている (特に、それぞれの構成要素が異なるコンテナーに含まれている場合や異なるシステム上に存在する場合)。		✓	✓	3.0
19.3	アプリケーションサーバーとデータベースサーバーの間など、構成要素間の通信が、必要最小限の権限を持つアカウントを使用して認証される。		✓	✓	3.0
19.4	アプリケーションの配備について、サンドボックス、コンテナー化、または隔離による防御が適切に行われており、攻撃者による他のアプリケーションの攻撃が遅延および阻止される。		✓	✓	3.0
19.5	アプリケーションの構築および配備プロセスが、セキュアな方法で行われることを確認する。		✓	✓	3.0
19.6	認可された管理者は、セキュリティ関連のすべての設定の完全性を検査し、設定が改ざんされていないことを確認する能力を保持している。			✓	3.0
19.7	すべてのアプリケーション構成要素が署名されている。			✓	3.0
19.8	サードパーティの構成要素は、信頼できるリポジトリからのものである。			✓	3.0

19.9	システムレベル言語でのビルドプロセスにおいて、ASLR、DEP、セキュリティチェックなど、すべてのセキュリティフラグが有効になっている。	✓	3.0

参照先

詳細については、以下を参照してください。

- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing: https://www.owasp.org/index.php/Testing_for_configuration_management

付録 A: 廃止された要件

我々に最も多く寄せられる質問は、「*…はどうなったのですか?*」という質問です。次の表に、廃止された要件がどうなったかを示します。既存の項目の変更や、新しい項目の追加については、この表ではなく、詳細セクション内で確認してください。

元の番号	説明	ステータス	削除	理由
2.3	許可された認証回数の上限を超えた場合に、ブルートフォース攻撃を防止できる十分な時間アカウントロックがかかる。	廃止	2.0	より複雑な要件 (v2.20) に置換。
2.5	外部認証サービスを利用しているライブラリを含むすべての認証の管理が集中実装されている。	統合	3.0	すべてのセキュリティ管理を含むように一般化され、1.10 に移動。
2.10	アプリケーション固有のセキュリティ上センシティブな処理を許可する前に、再度認証を要求する。	廃止	2.0	再認証はごくまれにしか見られないため、この管理策は削除することに決定した。
2.11	管理者が設定した期間を過ぎると、認証が確実に失効する。	廃止	2.0	絶対的なタイムアウトと資格情報の期限切れは有効な管理策ではないため削除。
2.14	外部サービスにアクセスするための認証情報は、暗号化され、ソースコードではなく保護された場所に保存されている。	更新	2.0	V2.21 へ更新。
2.15	認証の管理を実装するもしくは使用するコードが悪性コードの影響を受けない。	移動	2.0	「V13: 悪性コード」に移動。
3.8	セッション ID は再認証時に変更される。	更新	3.0	3.7 に統合。
3.9	ログアウト時にセッション ID が変更もしくは消去される。	更新	3.0	3.7 に統合。
3.13	実装された全ソースコードおよびセッション管理機能が悪性コードの影響を受けない。	移動	2.0	「V13: 悪性コード」に移動。
3.14	Cookie を使用した認証セッショントークンは "HttpOnly" の使用によって保護される。	更新	3.0	3.13 に移動。
3.15	Cookie を使用した認証セッショントークンは "secure" 属性によって保護される。	更新	3.0	3.13 に移動。

4.2	ユーザーは、認可されていない URL にアクセスできない。	更新	3.0	4.1 に統合。
4.3	ユーザーは、認可されていないデータファイルにアクセスできない。	更新	3.0	4.1 に統合。
4.13	業務上規定されている入力およびアクセス制限を超えることはできない (1 日の取引制限やタスクの順序変更など)。	移動	3.0	「V15: ビジネスロジック」に移動。
4.15	アクセス制御の機能を実装もしくは使用する全ソースコードが悪性コードの影響を受けない。	移動	2.0	「V13: 悪性コードの適切な処理」に移動。
5.2	すべての入力値に対してポジティブ検証が定義され、適用されている。	廃止	2.0	実装があまりに困難なため (特に自由形式のテキスト入力について) 削除。
5.4	すべての入力ソースについて文字セットを指定する (例: UTF-8)。	廃止	3.0	ほとんどの言語で実装があまりに困難なため削除。
5.7	すべての入力の検証の失敗は記録される。	廃止	3.0	あまりに多くの無駄なログが生成され、結果的に無視されるため削除。
5.8	すべての入力データが、検証前に、正規化された上でデコーダーやインタープリターに渡されている。	廃止	3.0	タイプ 1 JSP テクノロジ固有の要件であり、ほとんどの最新フレームワークにとって問題とはならないため削除。
5.9	すべての入力値検証が悪性コードの影響を受けない。	移動	2.0	「V13: 悪性コードの適切な処理」に移動。
5.14	ランタイム環境は XML 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によって XML インジェクションが防止される。	統合	3.0	V5.13 と統合。
5.15	-- 空要件 --	削除	3.0	存在しない要件。
5.19	アプリケーションによって行われる出力のエンコードやエスケープ処理の種類ごとに、単一のセキュリティ機構が存在する。	統合	3.0	すべてのセキュリティ制御を含むように一般化され、1.10 に移動。
7.1	ユーザーから送られた機密情報の暗号化はサーバー側に実装されている。	廃止	3.0	応答性の高い最新のアプリやモバイルアプリの多くに、この要件が計画的に組み込まれているため。
7.3	マスターシークレットに対する権限を持たない者からのアクセスは許可されない。マスターシークレットとは、アプリケーションの認証情報であり、ディスク上に平文で保存され、セキュリティ上の設定情報へのアクセスを制御するために用いられる。	移動	3.0	V2.29 に移動。

7.4	パスワードハッシュを生成する際には、ソルトを付加する。	移動	2.0	V2.13 に移動。
7.5	暗号化モジュールの失敗はログに保存する。	廃止	2.0	確認されることのない不要なログの作成は非生産的である。
7.10	暗号化モジュールをサポートまたは使用するすべてのコードが、悪性コードの影響を受けない。	移動	2.0	V13 に移動。
8.2	すべてのエラー処理は信頼できるデバイス上で実行される。		3.0	廃止
8.3	ログの保存に関する管理策がすべてサーバー上で実装されている。	移動	3.0	より包括的なアーキテクチャ管理策である V1.13 に移動。
8.9	アプリケーションレベルのログ記録は単一にする。	移動	3.0	より包括的なアーキテクチャ管理策である V1.13 に移動。
8.11	ログ記録のフォーマットの全項目で、ログイベントを複合検索条件で検索できるログ分析ツールを分析者が使用できるようにする。	廃止	3.0	セキュアなソフトウェアには不要であるため削除。
8.12	エラー処理とログの保存を実装するすべてのコードが、悪性コードの影響を受けない。	移動	2.0	「V13: 悪性コードの適切な処理」に移動。
8.15	トランザクションの実行前にログの保存が行われている。ログの保存に失敗した場合 (ディスクに空きがない、アクセス許可が不十分などの理由で) アプリケーションはフェイルセーフである。これは、完全性と否認不可が必須である場合の要件。	廃止	3.0	該当するアプリが全体のごく一部であり、あまりに細かい管理策であると考えられるため削除。
10.2	TLS 通信が失敗した場合に安全でない HTTP 接続にフォールバックしない。	統合	3.0	10.3 と統合。
10.7	センシティブな情報や機能を含む外部システムとの通信において、必要最小限の権限のみを持つように設定されたアカウントを使用する。			
10.9	すべての通信で特定の文字エンコーディング (UTF-8 など) を指定している。			
V11.1	廃止			
V11.4	廃止			
V11.5	廃止			
V11.6	廃止			
V11.7	廃止			
V11.8	廃止			

V11.4	廃止			
V13.1	廃止			
V13.2	廃止			
V13.3	廃止			
V13.4	廃止			
V13.5	廃止			
V13.6	廃止			
V13.7	廃止			
V13.8	廃止			
V13.9	廃止			
15.1 ～ 15.7、15.9	ビジネスロジックセクション。	統合	3.0	セクション 15 の大部分は 15.8 と 15.10 に統合。
15.11	STRIDE、つまりなりすまし (Spoofing)、改ざん (Tampering)、否認 (Repudiation)、情報漏えい (Information Disclosure)、権限昇格 (Elevation of privilege) に関連するリスクにアプリケーションが対処している。	重複	3.0	重複要件。V1.6 で取り込み済み。
16.4	ファイル名、パス名、またはあらゆるファイルシステムオブジェクトの使用時に、信頼できない情報源から取得したパラメーターが使用される場合、必ず事前に正規化および入力の検証が実行され、ローカルファイルインクルード攻撃が防止される。	移動	3.0	V16.2 に移動。
17.1	クライアントによって SSL 証明書が検証される。	廃止	3.0	重複要件。V10 で全般的な要件を取り込み済み。
V17.7	廃止			
V17.8	廃止			
V17.10	廃止			
V17.11	廃止			
V17.12	廃止			
V17.13	廃止			
V17.14	廃止			
V17.15	廃止			

V17.16	廃止			
V17.17	廃止			
V17.18	廃止			
V17.19	廃止			
V17.20	廃止			
V17.22	廃止			
V17.23	廃止			
V17.24	廃止			

付録 B: 用語集

- **アクセス制御 (Access Control)**: ファイル、機能、URL、データへのアクセスを、ユーザーが属するグループまたは ID によって制限する手段。
- **Address Space Layout Randomization (ASLR)**: バッファーオーバーフロー攻撃に対する防御に役立つ技法。
- **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、Open Systems Interconnection Reference Model (OSI Model) 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査 (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な評価。
- **アプリケーションセキュリティ検査報告書 (Application Security Verification Report)**: 特定のアプリケーションに対して検査者が行った分析と全体の結果を文書化した報告書。
- **認証 (Authentication)**: アプリケーションユーザーが表明する本人性の検査。
- **自動検査 (Automated Verification)**: 脆弱性のシグネチャを用いて問題を検出する自動ツール (動的分析、静的分析、その両方) を用いた検査。
- **バックドア (Back Doors)**: アプリケーションに承認されていないアクセスを許す悪性コードの一種。
- **ブラックリスト (Blacklist)**: 許可されていないデータまたは操作のリスト (入力データとして許可されていない文字のリストなど)。
- **カスケーディングスタイルシート (CSS: Cascading Style Sheets)**: HTML などのマークアップ言語で書かれたドキュメントのプレゼンテーションセマンティクスの記述に使用されるスタイルシート言語。
- **認証機関 (CA: Certificate Authority)**: デジタル証明書を発行するエンティティ。
- **通信セキュリティ (Communication Security)**: アプリケーション構成要素間、クライアントとサーバー間、外部システムとアプリケーション間でのデータの交信におけるデータ保護。
- **構成要素 (Component)**: 独立したコード単位。関連するディスク / ネットワークインターフェイスにより、他の構成要素と通信する。
- **クロスサイトスクリプティング (XSS: Cross-Site Scripting)**: 通常、Web アプリケーションで見られるセキュリティ脆弱性であり、コンテンツへのクライアント側スクリプトのインジェクションを許してしまう。
- **暗号化モジュール (Cryptographic module)**: 暗号化アルゴリズムの実装、暗号化鍵の生成を行う、ハードウェア、ソフトウェア、ファームウェア。

- **サービス拒否攻撃 (Denial of Service (DoS) Attacks)**: 処理能力を越えたリクエストを送信することでアプリケーションをオーバーフローさせる攻撃。
- **設計検査 (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。
- **動的検査 (Dynamic Verification)**: アプリケーションの実行中に脆弱性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した検査。
- **イースターエッグ (Easter Eggs)**: 悪性コードの一種で、ある特定の入力イベントが行われるまで実行されない。
- **外部システム (External Systems)**: アプリケーションの一部ではない、サーバーサイドアプリケーションまたはサービス。
- **FIPS 140-2**: 暗号化モジュールの設計および実装の検査基準として使用できる標準。
- **グローバル一意識別子 (GUID: Globally Unique Identifier)**: ソフトウェア内で識別子として使用される一意の参照番号。
- **ハイパーテキストマークアップ言語 (HTML: HyperText Markup Language)**: Web ブラウザーに表示される Web ページや他の情報の作成に使用される主要マークアップ言語。
- **ハイパーテキスト転送プロトコル (HTTP: Hyper Text Transfer Protocol)**: 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 のデータ通信の基盤。
- **入力検証 (Input Validation)**: 信頼できないユーザー入力を正規化および検証する。
- **ライトウェイトディレクトリアクセスプロトコル (LDAP: Lightweight Directory Access Protocol)**: ネットワークを介した分散ディレクトリ情報サービスへのアクセス / 管理に使用されるアプリケーションプロトコル。
- **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の知らない間にアプリケーション内に導入されたコードで、そのアプリケーションに設定されたセキュリティをかいくぐる。ウイルスやワームなどのマルウェアとは異なる。
- **マルウェア (Malware)**: アプリケーションに導入された実行コードで、アプリケーションの所有者や管理者に知られることなく実行される。
- **オープン Web アプリケーションセキュリティプロジェクト (OWASP: Open Web Application Security Project)**: 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の向上に重点的に取り組む、自由で公開されたコミュニティ。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" して、人々 / 団体がアプリケーションセキュリティのリスクに関して見識を持って決断を下すことができるようにすること。http://www.owasp.org/ を参照。
- **出力エンコード (Output encoding)**: Web ブラウザーや外部システムへのアプリケーションの出力を正規化および検証する。

- **個人識別情報 (PII: Personally Identifiable Information)**: 単独で、または他の情報と共に使用することで、個人の識別、個人への接触、個人の検出、あるいはコンテキストに応じた個人の識別ができる情報。
- **ポジティブ検証 (Positive validation)**: ホワイトリストを参照。
- **セキュリティアーキテクチャ (Security Architecture)**: セキュリティ管理の手法と実装箇所を記述、確認するアプリケーションのデザインの抽象化で、同時にアプリケーションとユーザーに関するデータの場所とその重要度も記述、確認する。
- **セキュリティ設定 (Security Configuration)**: セキュリティ制御の使用に影響を及ぼす、アプリケーションのランタイム設定。
- **セキュリティ管理 (Security Control)**: セキュリティチェックを実行する機能や構成要素 (アクセス制御チェックなど)、またはセキュリティに影響する結果を呼び出す機能や構成要素 (監査レコードの生成など)。
- **SQL インジェクション (SQLi: SQL Injection)**: データドリブンアプリケーションの攻撃に使用されるコードインジェクション技法。悪意のある SQL ステートメントがエントリポイントに挿入される。
- **静的検査 (Static Verification)**: アプリケーションのソースコードの問題点を脆弱性のシグネチャを用いて検知する自動化ツールを使用する検査。
- **検査対象 (TOV: Target of Verification)**: OWASP ASVS の要件に則ってアプリケーションセキュリティ検査を行う際に、検査対象となる特定のアプリケーション。このアプリケーションを "検査対象 (Target of Verification)" または TOV と呼ぶ。
- **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、セキュリティ制御、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を構築していくことからなる技法。
- **トランスポート層セキュリティ (Transport Layer Security)**: インターネットでの通信セキュリティを提供する暗号化プロトコル。
- **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。Uniform Resource Locator (URL) は、多くの場合、リソースの参照として使用される。
- **ユーザー受け入れテスト (UAT: User Acceptance Testing)**: 実稼働環境と同じように動作するテスト環境。ソフトウェアの実稼働化前に、あらゆるテストがこの環境で実施される。
- **検査者 (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。

- **ホワイトリスト (Whitelist)**: 許可されているデータまたは操作のリスト (入力検証で許可されている文字のリストなど)。
- **XML**: ドキュメントのエンコードに関するルールセットを定義するマークアップ言語。

付録 C: 参照先

この標準のユーザー / 採用者にとって役立つ可能性がきわめて高いものとして、以下の OWASP プロジェクトがあります。

- OWASP Testing Guide: https://www.owasp.org/index.php/OWASP_Testing_Project
- OWASP Code Review Guide: http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project
- OWASP Cheat Sheets: https://www.owasp.org/index.php/OWASP_Cheat_Sheet_Series
- OWASP Proactive Controls: https://www.owasp.org/index.php/OWASP_Proactive_Controls
- OWASP Top 10: https://www.owasp.org/index.php/Top_10_2013-Top_10
- OWASP Mobile Top 10: https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks

同様に、以下の Web サイトも、この標準のユーザー / 採用者にとって役立つ可能性がきわめて高いものです。

- MITRE Common Weakness Enumeration: http://cwe.mitre.org/
- PCI Security Standards Council: https://www.pcisecuritystandards.org
    - PCI Data Security Standard (DSS) v3.0 Requirements and Security Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf

付録 D: 標準の対応

PCI DSS 6.5 は、OWASP Top 10 2004/2007 から導出した内容をベースとし、これに、最近のプロセス拡張をいくつか追加したものです。ASVS は OWASP Top 10 2013 の厳密なスーパーセットであるため (OWASP Top 10 の 10 項目に対して、ASVS は 154 項目)、OWASP Top 10 と PCI DSS 6.5.x で扱われている問題のすべてが、より詳細な ASVS 制御要件で処理されています。たとえば、「破損のある認証およびセッション管理」は、「V2: 認証」セクションおよび「V3: セッション管理」セクションに正確に対応しています。

検査レベル 3 では、完全な対応が達成されます。一方、検査レベル 2 では、PCI DSS 6.5 のほとんどの要件に対処できますが、6.5.3 と 6.5.4 は含まれません。

PCI DSS 6.5.6 などのプロセスの問題は、ASVS では扱っていません。

PCI-DSS 3.0	ASVS 3.0	説明
6.5.1 インジェクションの欠陥 (特に、SQL インジェクション)。また、OS コマンドインジェクション、LDAP および XPath インジェクション、および他のインジェクションの欠陥についても考慮すること。	5.11、5.12、5.13、8.14、16.2	正確に対応。
6.5.2 バッファーオーバーフロー	5.1	正確に対応。
6.5.3 安全性の低い暗号ストレージ	v7 のすべて	レベル 1 以上で包括的に対応。
6.5.4 安全性の低い通信	v10 のすべて	レベル 1 以上で包括的に対応。
6.5.5 不適切なエラー処理	3.6、7.2、8.1、8.2	正確に対応。
6.5.7 クロスサイトスクリプティング (XSS)	5.16、5.20、5.21、5.24、5.25、5.26、5.27、11.4、11.15	ASVS では、特にレガシアプリケーションでの XSS に対する防御の複雑性に重点をおいて、いくつかの要件に XSS を分類している。

6.5.8 不適切なアクセス制御 (安全性の低い直接的なオブジェクト参照、URL へのアクセス制限の失敗、ディレクトリトラバーサル、機能へのユーザーアクセス制限の失敗など)	v4 のすべて	レベル 1 以上で包括的に対応。
6.5.9 クロスサイトリクエストフォージェリ (CSRF)	4.13	正確に対応。ASVS では、CSRF に対する防御をアクセス制御の側面の 1 つと見なしている。
6.5.10 欠陥のある認証およびセッション管理	v2 と v3 のすべて	レベル 1 以上で包括的に対応。