OWASP

Open Web Application Security Project

Application Security Verification Standard 3.0.1

2015 年 12 月

謝辞 6

本標準について 6
著作権とライセンス 6
バージョン 3.0 (2015) 6
バージョン 2.0 (2014) 7
バージョン 1.0 (2009) 7

序文 9

バージョン 3.0 の更新内容 9

APPLICATION SECURITY VERIFICATION STANDARD の使用 11

アプリケーションセキュリティの検証レベル 12
本標準の採用方法 12
レベル 1: 導入 (OPPORTUNISTIC) 12
レベル 2: 標準 (STANDARD) 13
レベル 3: 上級 (ADVANCED) 13
ASVS 適用の実際 14

採用事例 17

採用事例 1: セキュリティテストの指針として 17
採用事例 2: セキュアな開発ライフサイクルとして 18

検証レベルの達成を評価する 20

ASVS 認定と信頼マークに対する OWASP の見解 20
認定組織に対する指針 20
自動ペネトレーションテストツールの役割 21
ペネトレーションテストの役割 21
セキュリティアーキテクチャの詳細な指針として 21
既存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 21
自動の単体および統合テストのガイドとして 22
セキュア開発のためのトレーニングとして 22

ASVS を採用している OWASP のプロジェクト 23

SECURITY KNOWLEDGE FRAMEWORK 23
OWASP ZED ATTACK PROXY 23
OWASP CORNUCOPIA 23

検証要件の詳細 24

V1: アーキテクチャ，設計，脅威モデリング 25

管理目標 25

要件 25
参考情報 26

V2: 認証に関する検証要件 27

管理目標 27
要件 27
参考情報 29

V3: セッション管理に関する検証要件 30

管理目標 30
要件 30
参考情報 31

V4: アクセス制御に関する検査要件 32

管理目標 32
要件 32
参考情報 33

V5: 悪性入力の処理に関する検証要件 34

管理目標 34
要件 34
参考情報 36

V6: 出力のエンコード / エスケープ 38

V7: 暗号化に関する検証要件 39

管理目標 39
要件 39
参考情報 40

V8: エラー処理とログの保存に関する検証要件 41

管理目標 41
要件 41
参考情報 42

V9: データの保護に関する検証要件 43

管理目標 43
要件 43
参考情報 45

V10: 通信のセキュリティに関する検証要件 46

管理目標 46
要件 46
参考情報 47

V11: HTTP のセキュリティ設定に関する検証要件 49

管理目標 49
要件 49
参考情報 50

V12: セキュリティ設計に関する検証要件 51

V13: 悪性活動の管理に関する検証要件 52

管理目標 52
要件 52
参考情報 52

V14: 内部セキュリティに関する検証要件 53

V15: ビジネスロジックに関する検証要件 54

管理目標 54
要件 54
参考情報 54

V16: ファイルとリソースに関する検証要件 56

管理目標 56
要件 56
参考情報 57

V17: モバイルに関する検証要件 58

管理目標 58
要件 58
参考情報 59

V18: WEB サービスに関する検証要件 60

管理目標 60
要件 60
参考情報 61

V19. 構成 62

管理目標 62
要件 62

参考情報 63

付録 A: 廃止された要件 64

付録 B: 用語集 70

付録 C: 参考情報 74

付録 D: 他の標準との対応 75

謝辞

本標準について

Application Security Verification Standard（アプリケーションセキュリティ検証標準）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，ユーザは，本標準を使用することでセキュアなアプリケーションとは何かを定義することができます．

日本語版について

本書（バージョン 3.0.1）は経済産業省の委託事業として一般社団法人 JPCERT コーディネーションセンターが翻訳したものです．

日本語版の内容について，原著に沿ってできるだけ忠実に翻訳するよう努めていますが，完全性，忠実性を保証するものではありません．また，邦訳者は本文書に記載されている情報により生じる損失または損害に対し，いかなる人物あるいは団体にも責任を負うものではありません．

著作権とライセンス

バージョン 3.0 (2015)

プロジェクトリーダー	主執筆者	共同執筆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)

プロジェクトリーダー	主執筆者	共同執筆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)

プロジェクトリーダー	主執筆者	共同執筆者およびレビュー担当者
Mike Boberski Jeff Williams Dave Wichers	Jim Manico	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

序文

Application Security Verification Standard (ASVS) バージョン 3.0 へようこそ．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策の標準化に重点をおいています．

ASVS v3.0 は，コミュニティにおける取組みと業界からのフィードバックの成果です．本リリースでは，ASVS の採用に関する事例を示すことが重要であると考えました．事例をみることで，ASVS に初めて接する企業，本標準の採用計画を容易に行えるようになり，既に採用している企業は他の採用企業の経験から学ぶことができるでしょう．

本標準の内容に 100% 同意いただけるとは我々も考えていません．リスク分析には主観が常につきものです．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，尊重しつつもさらに強化することを期待しています．

バージョン 3.0 の更新内容

バージョン 3.0 では，最新なアプリケーションへの本標準の適用可能性を強化するため，構成，Web サービス，モダンな (クライアントベースの) アプリケーションなどに関する新たなセクションを追加しました．モダンなアプリケーションとは，一般に応答性を備えるアプリケーションであり，HTML5 を活用したフロントエンドやモバイルクライアントから，SAML 認証を使用して共通の RESTful Web サービス群を呼び出すようなものを想定しています．

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，脆弱性の悪用可能性や悪用され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，脆弱性の影響をどのように軽減できるかの洞察を得る助けとなるということです．

これまで我々は，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，コミュニティのフィードバックを大量に取り込んできました．ピアレビュー時に管理策の意味が大幅に変更されたときは，新たな管理策を作成し，既存の管理策は廃止しました．廃止した管理策は，混乱を招く可能性が

あるため，あえて再利用しないこととしました．変更内容は，付録 A に包括的な対応表として収録しています．

まとめると，v3.0 ではこれまでで最も大きな変更が行われています．今回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 つの主目的があります.

- 組織におけるセキュアなアプリケーション開発と保守を支援する
- セキュリティサービス，セキュリティツールベンダ，およびユーザが，製品やサービスをセキュリティ要件に合致させる

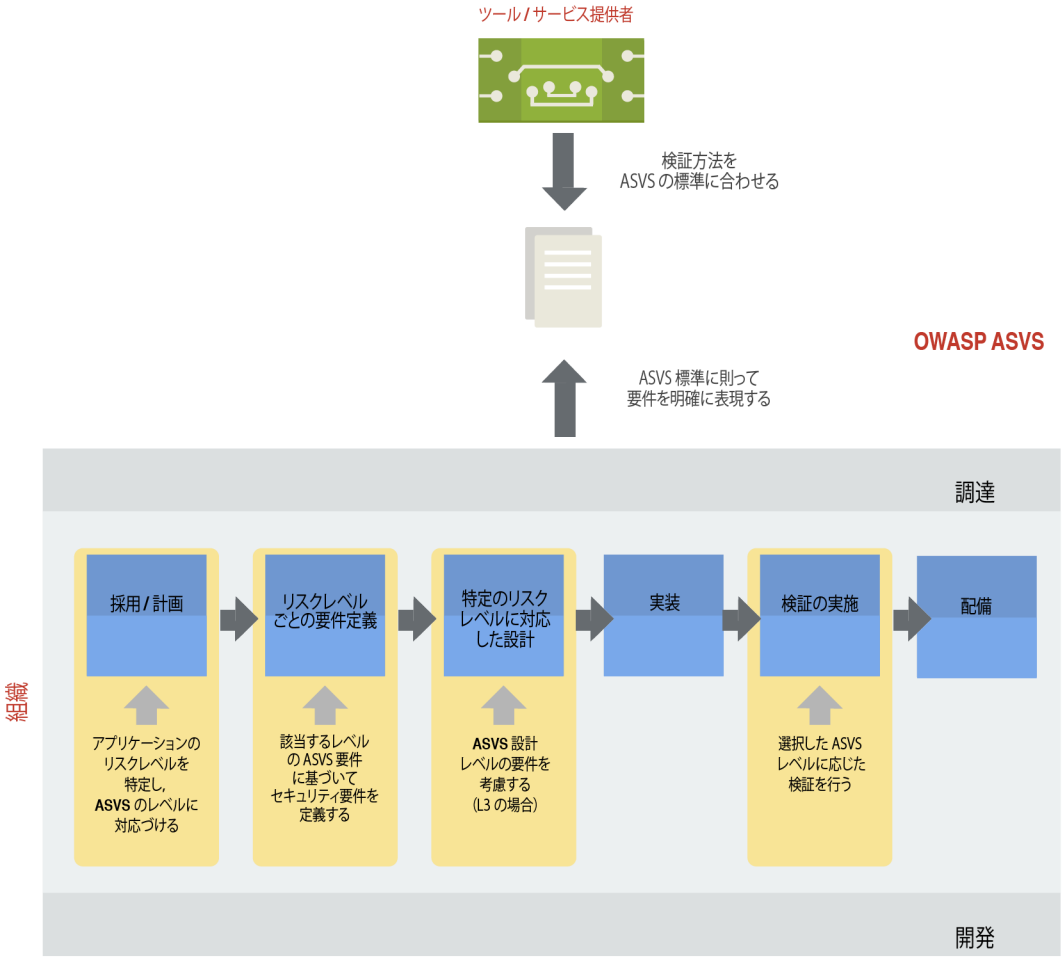

図 1 – 組織およびツール/サービス提供者における ASVS の活用

アプリケーションセキュリティの検証レベル

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，レベルが上がる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する
- ASVS レベル 2 は，保護を必要とする機微なデータを含むアプリケーションを対象とする
- ASVS レベル 3 は，最も重要度が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，機微な医療データを含むものなど）を対象とする

各 ASVS レベルでは，セキュリティ要件を列挙しています．また，個々の要件は，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．

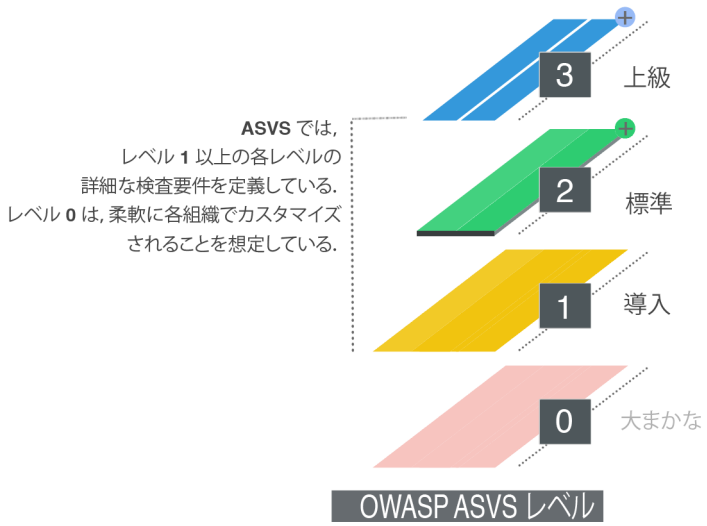

図 2 - OWASP Application Security Verification Standard 3.0 のレベル

本標準の採用方法

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．ユースケースに合うように ASVS を変更することで，プロジェクトや環境で最も重要なセキュリティ要件により注力することができます．

レベル 1: 導入 (Opportunistic)

OWASP Top 10 や同類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する防御が適切に行われているアプリケーションは，ASVS レベル 1 (導入) を満たします．

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．また，エンタープライズアプリケーションの簡易分析のためや，複数のフェーズからなる活動の 1 つとして，セキュリティ要件の優先順位リストの作成を補助するために適しています．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，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の多くは，単純で容易に使える技術を用い，簡単に発見できて悪用が容易な脆弱性を発見する攻撃者に由来するものです．これとは対照的に，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．アプリケーションで価値の高い重要なデータを処理している場合，レベル 1 では不十分でしょう．

レベル 2: 標準 (Standard)

今日のソフトウェアが直面するほとんどのリスクに適切に対応するアプリケーションは，ASVS レベル 2 (標準) を満たします．

レベル 2 では，アプリケーション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微な機能を実装する，その他の機微な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，攻撃者は，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い，特定の標的を攻撃します．

レベル 3: 上級 (Advanced)

レベル 3 は ASVS で最も高い検査レベルです．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後段に示します．高度な脆弱性に対する対策が適切に行われた，正しいセキュリティ設計の原則を実践しているアプリケーションは，ASVS レベル 3 (上級) を満たします．

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です．セキュアなアプリケーションは，(弾力性やスケーラビリティの確保，また多層的セキュリティの実現のために)適切にモジュール化されており，(ネットワーク接続によって，または物理的に分離された)各モジュールは与えられたセキュリティ上の役割を果たします（多層防御)．また，各モジュールの役割は，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(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 (システム間の認証など)，否認防止，認可，および監査 (ログの取得) などの確保があります．

ASVS 適用の実際

脅威が異なればその動機も異なります．業界によっては，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．

以下の表に，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，脅威の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次の問題があります．つまり，日和見的な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 が業界を問わずあらゆるアプリケーションに推奨されるのはそのためであり，容易に見つかるリスクを管理する出発点として推奨されます．組織は，事業の性質に基づいて，自社に固有のリスク特性をより綿密に調査することが強く求められます．対極にあるのが ASVS レベル 3 です．レベル 3 は，生命の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	業界のプロファイル	L1 推奨対象	L2 推奨対象	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．

業界	業界のプロファイル	L1 推奨対象	L2 推奨対象	L3 推奨対象
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． 攻撃者の大半は，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．	ネットワークアクセス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．
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．	ネットワークアクセス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．

業界	業界のプロファイル	L1 推奨対象	L2 推奨対象	L3 推奨対象
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．

採用事例

採用事例 1: セキュリティテストの指針として

米国ユタ州のある私立大学では，学内の Red Team が，アプリケーションの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，ペネトレーションテストのプロセス全体（最初の計画と対象検討のためのミーティングからテストの実施指針まで）において，また，クライアント向け最終報告書にテスト結果まとめるための枠組みとして，使用されています．また，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．最初の計画ミーティングの際，クライアントは多くの場合，学生チームがアプリケーションをテストすることに後ろ向きです．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．次に，テスト範囲の検討において，ASVS を使ってテストに要する時間と作業が見積もられます．Red Team は，あらかじめ定義された ASVS の検査レベルを使い，リスクに基づいたテストが行われることを説明します．ASVS の活用により，クライアント，関係者，そして Red Team はテストの対象となるアプリケーションの適切なスコープについて合意します．

ひとたびテストが始まると，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，各検証要件のテストの状況 (完了，保留中など) を追跡することで，テストの進捗状況を容易に把握できます．これにより，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，プロジェクトマネージャーはより適切にリソースを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，チームメンバーの大半は複数の講義に時間を割かなくてはなりません．しかし，個々の検証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．報告書作成も，ASVS は構成が明確であるため，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合わせて報告書の大半を効果的に作成することができます．

Red Team は ASVS に沿って最終報告書を構成し，各検証要件のステータスを報告し，必要に応じて詳細を提供します．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，あとに続く取組みにも非常に役立ちます．さらに，報告書形式が 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，その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の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，個々の検証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．

採用事例 2: セキュアな開発ライフサイクルとして

金融機関向けにビッグデータ分析機能の提供を目論む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 を採用しました．

スタートアップ企業は ASVS を使用し，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の ASVS の活用方法は通常とは異なるものでした．まず 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，TOTP 2 要素認証の追加を選択し，同時にパスワードポリシーとブルートフォースの検出および防止機構の効果を倍増させる Web サービスレギュレータを追加するという具合です．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 を使って，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をチェックします．

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．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，単一の保証に頼るのは不十分です．テスト体制を自動化することで，熟練したペネトレーションテスト担当者が数週間テストしても重大な問題は見つからないでしょう．

検証レベルの達成を評価する

ASVS 認定と信頼マークに対する OWASP の見解

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．

ASVS に関する保証の表明，信頼マーク，認定は，いずれも OWASP によって正式に検査や登録，または認定されたものではありません．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．

認定組織に対する指針

Application Security Verification Standard は，アプリケーションの公開検証にも活用することができます．公開検証においては，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なコードレビューでは問題を例外として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現れるというわけです．認定機関は，報告書に，検証の範囲 (特に，SSO 認証などの重要な構成要素が範囲外である場合)，検査結果の要約（合格したテストと不合格だったテスト），および不合格のテストに対する解決策の明確な指示を必ず含める必要があります．

詳細な作業文書，スクリーンショットや動画，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，プロキシのログや，クリーンアップリストなどの関連するメモ) 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証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報告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と裏付ける十分な証明には (まったく) なりません．論争が発生したときは，検証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

ソースコードにアクセスせず，手動によるペネトレーションテストを通じて L1 のすべての問題を検査することは可能ですが，主流の手法ではありません．L2 では，開発者，ドキュメント，コードへの一定のアクセスと，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をカバーするのは不可能です．なぜなら，レベル 3 でカバーされる問題の大半が，システム構成のレビュー，悪性コードのレビュー，脅威モデリングといったペネトレーションテスト以外の成果物を必要とするからです．

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がリソースとして使用することです．セキュリティアーキテクチャフレームワークとして代表的な 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をレビュー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とのギャップを埋めることができます．

既存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 を採用することでメリットを得られるでしょう．採用の方法としては， 3 つのレベルのいずれかを選択する方法もあれば， ASVS をフォークし，各アプリケーションリスクレベルで必要とされる要件を分野に合った形に改変するという方法もあるでしょう．この種のフォークは奨励されますが，前提条件として，追跡可能性が維持される必要があります．

つまり，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，フォークが進んでいっても，フォークしたコピーが ASVS 標準と同じ要件を満たすことがわかります．

自動の単体および統合テストのガイドとして

ASVS は，アーキテクチャに関する要件と悪性コードに関する要件を除き，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ズデータ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証可能になります．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，よくあるパスワード，パスワード長，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が含まれるべきでしょう．

セキュア開発のためのトレーニングとして

ASVS はセキュアなソフトウェアの特性を定義するためにも使用できます．"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らないでしょう．やってはいけないトップ 10 項目を取り上げるのではなく，ASVS の事前対処の管理策により重点をおいたコースにするとよいでしょう．

ASVS を採用している OWASP のプロジェクト

Security Knowledge Framework

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

開発者のセキュアコーディングトレーニング．SKF はオープンソースの Python-Flask Web アプリケーションで，OWASP Application Security Verification Standard を使用し，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もの．

OWASP Zed Attack Proxy

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

OWASP Zed Attack Proxy (ZAP) は，Web アプリケーションの脆弱性を検出するための使い勝手の良い統合ペネトレーションテストツールです．ZAP は，セキュリティの経験が様々なユーザが使うことを想定して設計されており，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，自動スキャナに加え，脆弱性を手動で検出するための一連のツールを備えている．

OWASP Cornucopia

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

OWASP Cornucopia は，カードゲームの形式で，ソフトウェア開発チームがアジャイル，従来の，あるいは形式的開発プロセスにけるセキュリティ要件の設定を助けるものです．ゲームは開発言語やプラットフォーム，テクノロジに依存しません．Cornucopia のスーツ（トランプのマーク）は，OWASP Secure Coding Practices - Quick Reference Guide (SCP) の構成に基づいて選ばれていますが，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．

検証要件の詳細

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. 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. 悪性活動の管理

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 で追加)

V19. 構成 (3.0 で追加)

V1: アーキテクチャ，設計，脅威モデリング

管理目標

検証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を把握しており，各構成要素に存在理由がある
- レベル 2: アーキテクチャが定義されており，コードがアーキテクチャに準拠している
- 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

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の管理策と同じ内容です．

要件

#	説明	1	2	3	導入
1.1	アプリケーションのすべての構成要素を把握し，それらが必要とされている	✓	✓	✓	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その動作に必要な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の観点で定義されている			✓	1.0
1.5	アプリケーションに内包されては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の観点で定義されている			✓	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)，情報漏洩 (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策（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		✓	✓	3.0
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

#	説明	1	2	3	導入
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分離されており，信頼されたシステム上でセキュリティに関する意思決定を適用できる		✓	✓	3.0
1.10	クライアント側のコードに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含まれていない		✓	✓	3.0

参考情報

詳しくは以下の情報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: 認証に関する検証要件

管理目標

認証 (Authentication) とは，対象が本物(あるいは本人) であることを立証もしくは確認する行為です．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．検証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 を検証する
- 許可された者だけが認証でき，認証情報がセキュアな方法で転送される

要件

#	説明	1	2	3	導入
2.1	意図して公開しているものを除き，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，ユーザが入力したパスワードがそのまま表示されない	✓	✓	✓	1.0
2.4	すべての認証がサーバ側で行われる	✓	✓	✓	1.0
2.6	すべての認証機構がフェイルセキュアであり，攻撃者がログインできない	✓	✓	✓	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 (プロファイルの更新機能，パスワードの再設定，トークンの無効化や紛失，ヘルプデスク，IVR など) が，一義的な認証機構として攻撃耐性を持つ	✓	✓	✓	2.0
2.9	パスワード変更機能に，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	疑わしい認証成功と失敗がすべてログに保存されている．ログには，セキュリティ調査に必要な関連するメタデータを含むリクエストが含まれていること		✓	✓	2.0
2.13	アカウントパスワードに十分な強度を持つ暗号化ルーチンを使われており，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

#	説明	1	2	3	導入
2.16	認証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	パスワード再設定機能やその他のパスワード回復経路から現在のパスワードが露呈せず，新しいパスワードが平文でユーザに送信されない	✓	✓	✓	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ってアカウント情報の取得はできない	✓	✓	✓	2.0
2.19	アプリケーションのフレームワークやアプリケーションが使用するすべての構成要素でデフォルトパスワード ("admin/password" など) を使用しない	✓	✓	✓	2.0
2.20	ブルートフォース攻撃や DoS 攻撃など認証機構に対する一般的な自動攻撃を防止するために，リクエスト制限機能が存在する	✓	✓	✓	3.0
2.21	外部サービスにアクセスするための認証証明がすべて暗号化されており，安全な場所に保管されている		✓	✓	2.0
2.22	パスワードの再設定やその他のパスワード復帰方法において，ソフトトークン，モバイルプッシュ，またはオフラインのパスワード回復機構が使用されている	✓	✓	✓	3.0
2.23	アカウントロックが，ソフトロック状態とハードロック状態に分かれていて，相互に排他的でない．つまり，ブルートフォース攻撃によりアカウントが一時的にソフトロックされた場合，ハードロック状態はリセットされない		✓	✓	3.0
2.24	知識に基づく質問 ("秘密の質問") を実装する必要がある場合，アプリケーションを保護できる十分強固な質問である	✓	✓	✓	2.0
2.25	過去に使用されたパスワードの再使用禁止について，何世代前までを禁止するかシステムで設定できる		✓	✓	2.0
2.26	アプリケーション固有のセンシティブな処理がアプリケーションのリスク特性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	よくある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とられている	✓	✓	✓	3.0
2.28	成功，失敗にかかわらず，認証要求に対する応答が同じ平均応答時間で返される			✓	3.0

#	説明	1	2	3	導入
2.29	シークレット，API 鍵，パスワードはソースコードやオンラインのソースコードリポジトリ中に含まれていない			✓	3.0
2.30	ユーザが認証を行うことをアプリケーションが許可する場合，安全性が実証されている認証メカニズムを使用する	✓	✓	✓	3.0
2.31	ユーザが認証を行うことをアプリケーションが許可する場合，ユーザは，ユーザ名とパスワードの漏洩に対策するために，2 要素認証や別の強力な認証等の機構用いることができる		✓	✓	3.0
2.32	信頼できない第三者が管理インターフェイスにアクセスできない	✓	✓	✓	3.0

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 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 at
  https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet

V3: セッション管理に関する検証要件

管理目標

Web アプリケーションの中核を構成する要素の 1 つは，アプリケーションと対話するユーザの状態を管理・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，ユーザと Web アプリケーション間のステートフルな対話を制御する管理策として定義されます．

検証対象のアプリケーションが次の高次のセッション管理に関する要件を満たすことを確認します．

- セッションがユーザ毎に固有であり，推測や共有ができない
- セッションは，不要になると無効になり，非アクティブ状態が続くとタイムアウトする

要件

#	説明	1	2	3	導入
3.1	セッションマネージャーを独自実装していない，または独自実装のセッションマネージャーはあらゆる一般的セッション管理攻撃に耐えられる	✓	✓	✓	1.0
3.2	ユーザがログアウトするとセッションが無効になる	✓	✓	✓	1.0
3.3	一定期間非アクティブ状態が続くとセッションがタイムアウトする	✓	✓	✓	1.0
3.4	管理者が設定可能な上限時間を超えると，ユーザのアクティビティに関係なくセッションがタイムアウトになる(絶対タイムアウト機能)			✓	1.0
3.5	認証を必要とするすべてのページに，見やすくて分かりやすいログアウト機能がある	✓	✓	✓	1.0
3.6	セッション ID が URL やエラーメッセージ，ログを通じて漏洩しない．また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	認証と再認証の成功時に必ず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されたセッション ID のみがアクティブであると認識される		✓	✓	1.0

3.11	セッション ID が十分な長さを持ち，ランダムであり，かつアクティブである正しいセッションベース全体で一意である	✓	✓	✓	1.0
3.12	Cookie に保存されたセッション ID において，アプリケーションのみに限定されたパスが適切に設定されており，さらに，認証セッショントークンには"HttpOnly" 属性と "secure" 属性が設定されている	✓	✓	✓	3.0
3.16	同時にアクティブなセッション数を制限している	✓	✓	✓	3.0
3.17	ユーザ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，アクティブなセッションの一覧が表示される．ユーザは任意のアクティブなセッションを終了することができる	✓	✓	✓	3.0
3.18	ユーザがパスワード変更に成功すると，他のアクティブなセッションをすべて終了するかどうかの確認が求められる	✓	✓	✓	3.0

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
  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 (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が有効な認証情報を持つ
- 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
- ロールとアクセス許可のメタデータがリプレイや改ざんから保護されている

要件

#	説明	1	2	3	導入
4.1	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，なりすましや権限昇格に対する防御となる	✓	✓	✓	1.0
4.4	センシティブなレコードへのアクセスが保護されており，ユーザは，認可されたオブジェクトやデータのみにアクセスできる (たとえば，ユーザがパラメーターを改ざんして他のユーザのアカウント情報を表示したり変更することを防止する)	✓	✓	✓	1.0
4.5	ディレクトリリスティグは，意図して許可されない限り，無効である．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 フォルダーなど) の検出や開示が許可されていない	✓	✓	✓	1.0
4.8	アクセス制御がフェイルセキュアである	✓	✓	✓	1.0
4.9	プレゼンテーション層と同じアクセス制御のルールがサーバー側に適用されている	✓	✓	✓	1.0
4.10	ユーザは，特に許可されない限り，アクセス制御に使用されるユーザ属性，データ属性，ポリシー情報を変更できない		✓	✓	1.0
4.11	保護された各リソースへのアクセスを保護するための集中管理メカニズム(外部の認可サービスを呼び出すライブラリを含む)を備えている			✓	1.0
4.12	アクセス制御上の決定をすべてログに保存することができ，すべてのアクセス失敗がログに記録されている		✓	✓	2.0

#	説明	1	2	3	導入
4.13	アプリケーションまたはフレームワークが，強固なランダムのアンチ CSRF トークンを使用するか，別のトランザクション保護メカニズムを備えている	✓	✓	✓	2.0
4.14	保護された機能やリソース，データに対する集中的もしくは連続的アクセスからシステムを守ることができる．たとえば，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，単一ユーザ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)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		✓	✓	3.0
4.16	アプリケーションが適切に状況に応じた認可を行っており，パラメーターの改ざんによる，認可されていない操作が不可能である	✓	✓	✓	3.0

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
  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証要件

管理目標

最もよく作り込まれる Web アプリケーションのセキュリティ上の弱点は，クライアントまたは環境から受信した入力を検証しないまま使用するというものです．この弱点は，クロスサイトスクリプティングや SQL インジェクション，インタープリタインジェクション，ロケールや Unicode を使った攻撃，ファイルシステムに対する攻撃，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，本来の目的に合致していることを確認している
- 外部エンティティやクライアントから取得したデータは，決して信頼せず，適切に扱う

要件

#	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ない，もしくはバッファーオーバーフローを防止する対策を行っている	✓	✓	✓	1.0
5.3	サーバー側の入力検証で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	入力値検証のルーチンがサーバー側で実施されている	✓	✓	✓	1.0
5.6	アプリケーションに許可されるデータの検証において，各データ型毎に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およびストアドプロシージャ呼出しが，プリペアドステートメントを使用している，もしくはクエリのパラメーター化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	✓	✓	✓	2.0
5.12	アプリケーションが 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管理策によって OS コマンドインジェクションが防止される	✓	✓	✓	2.0

#	説明	1	2	3	導入
5.13	ファイルパスがコンテンツとして使用されたときに，アプリケーションが，リモートファイルインクルード (RFI) や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	アプリケーションが 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よくある XML 攻撃の影響を受けない	✓	✓	✓	2.0
5.15	HTML や他の Web クライアントコード中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る，もしくはコンテキストに応じて自動的にエンコードを行うテンプレートを使用しており，アプリケーションが，反射型，格納型，および DOM 型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において受信リクエストからモデルへのパラメーターの自動一括割り当て (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センシティブなフィールドが，悪意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 変数汚染攻撃に対する防御策を備えている．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のソース(GET，POST，Cookie，ヘッダー，環境など)を区別しない場合は特にこの防御が必要		✓	✓	2.0
5.18	サーバー側の入力値検証に加え，第 2 の防衛線として，クライアント側の入力値検証が行われている		✓	✓	3.0
5.19	すべての入力データが検証されている．入力データには，HTML のフォームのフィールドだけでなく，REST 呼び出しやクエリパラメーター，HTTP ヘッダ，Cookie，バッチファイル，RSS フィードなども含む．検証はホワイトリスト方式で行う．それができない場合は，グレーリスト方式 (入力に含まれる既知の不適切な文字列を排除) あるいはブラックリスト方式（不適切な入力全体を拒否）で行う．		✓	✓	3.0
5.20	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検証される．（例：クレジットカード番号や電話番号などの検証や，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であることの検証）		✓	✓	3.0
5.21	非構造化データが無害化され，使用可能な文字と長さ等一般的な対策が適用されている．また，特定のコンテキストで有害となりうる文字 (たとえば，"ねこ" や "O'Hara" など，Unicode 文字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

#	説明	1	2	3	導入
5.22	WYSIWYG エディタ等から取得した信頼できない HTML が，HTML サニタイザーによって適切に無害化され，入力検証やエンコードにより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ときは，代わりに HTML の無害化が有効である		✓	✓	3.0
5.24	1 つの DOM コンテキストから別の DOM コンテキストへデータを転送する際には安全な JavaScript メソッドが使用されている（.innerText や .val の使用など）		✓	✓	3.0
5.25	ブラウザで JSON をパースするときは JSON.parse を使用する．eval() を使用しない．		✓	✓	3.0
5.26	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		✓	✓	3.0

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
  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は次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

本セクションは Application Security Verification Standard 2.0 の要件 V5 に取り込まれました．ASVS の要件 5.16 では，クロスサイトスクリプティング対策としてコンテキストに応じた出力のエンコードに触れています．

V7: 暗号化に関する検証要件

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において，暗号化に失敗した場合の安全対策を施しており，エラー処理が適切に行われる
- ランダム性が必要とされる場合，適切な乱数生成器を使用する
- 鍵に対するアクセスが安全な方法で管理されている

要件

#	説明	1	2	3	導入
7.2	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	✓	✓	✓	1.0
7.6	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		✓	✓	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がFIPS 140-2 または同等の標準で検証されている	✓	✓	✓	1.0
7.8	暗号化モジュールが，セキュリティポリシーに従い，許可されたモードで動作している			✓	1.0
7.9	暗号鍵の管理方法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），適切に運用されている		✓	✓	1.0
7.11	暗号化サービスを使うすべての処理は，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の扱いなど，暗号化関連のプロセスを複数に分離するとともに，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	個人識別情報が暗号化された上で保存されて保護されたチャネルを介してやりとりされる		✓	✓	3.0
7.13	可能な限り，鍵およびシークレットを破棄する際にゼロで上書きする		✓	✓	3.0

#	説明	1	2	3	導入
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もしくは置換される		✓	✓	3.0
7.15	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フェイルセーフな動作をする			✓	3.0

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
  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証要件

管理目標

エラー処理とログ保存の主な目標は，ユーザ，管理者，およびインシデント対応チームに有益な対応を行う手段を提供することです．したがって，大量のログを生成することが目的ではなく，ノイズよりもシグナルが多い，高品質のログを生成することが目的です．

高品質のログには，多くの場合，センシティブなデータが含まれるため，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．これには次が含まれます．

- 必要な場合を除き，センシティブな情報の収集やログの保存を行わない
- ログに保存されたすべての情報が，データの分類に基づいてセキュアな方法で処理され，保護される
- ログを永続的に保存せず，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センシティブなデータ (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り，攻撃者にとってきわめて魅力的なものとなります．

要件

#	説明	1	2	3	導入
8.1	攻撃者に利用される可能性があるセンシティブなデータ (セッション ID，ソフトウェアやフレームワークのバージョン，個人情報など) をエラーメッセージやスタックトレースに表示しない	✓	✓	✓	1.0
8.2	セキュリティ管理策においてエラー処理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	セキュリティへの影響が考えられるイベントについて，成功，失敗両方のログを保存できる管理策を用意する		✓	✓	1.0
8.4	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		✓	✓	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不正なアクセスや改変から保護されている		✓	✓	1.0
8.7	プライバシー法や規制で定義されているセンシティブなデータ，リスク評価で定義されている組織のセンシティブなデータ，あるいは攻撃者に利用される可能性があるセンシティブな認証データ (ユーザのセッション ID，パスワード，ハッシュ，API トークンなど) を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，ログのエントリの中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ている			✓	2.0
8.9	ログエントリ中で，信頼できる情報源から取得したフィールドと信頼できない情報源から取得したフィールドを区別できる			✓	2.0
8.10	監査ログまたは同様の存在によって，主要トランザクションの否認不可が実現されている		✓	✓	3.0
8.11	セキュリティログに何らかの完全性チェック機構が備わっており，認可なくログを変更できない			✓	3.0
8.12	アプリケーションが動作する場所とは異なるパーティションにログが保存され，適切なログローテーションが行われている			✓	3.0

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

V9: データの保護に関する検証要件

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．共有コンピューターや電話，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

要件

#	説明	1	2	3	導入
9.1	センシティブな情報を扱うすべてのフォームにおいてクライアントサイドのキャッシュ (オートコンプリート機能を含む) を許可しない	✓	✓	✓	1.0
9.2	アプリケーションが処理するセンシティブなデータがリスト化されており，関連するデータ保護条令の元で，これらデータに対するアクセスの制限，暗号化，適用を定めるポリシが明文化されている			✓	1.0
9.3	すべてのセンシティブなデータが HTTP メッセージの本文またはヘッダによってサーバーに送信されている（URL パラメーターを使用したデータ送信を行わない）	✓	✓	✓	1.0

#	説明	1	2	3	導入

9.4	アプリケーションが抱えるリスクに応じてキャッシュを禁止するヘッダを設定する．例を次に示す．
	`Expires: Tue, 03 Jul 2001 06:00:00 GMT`
	`Last-Modified: {now} GMT`
	`Cache-Control: no-store, no-cache, must-revalidate, max-age=0`
	`Cache-Control: post-check=0, pre-check=0`
	`Pragma: no-cache`
		✓	✓	✓	1.0

9.5	サーバー上のセンシティブなデータのキャッシュや一時コピーが許可のないアクセスから保護されている，または，許可されたユーザがアクセスした後に無効化または削除される		✓	✓	1.0
9.6	データ保持ポリシーで定める期間が終了した時に各種のセンシティブな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	1 リクエスト中に含まれるパラメーターの数（非表示フィールド，Ajax 変数，Cookie，ヘッダー値など）を最小限にしている		✓	✓	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等データ収集が目的の異常な数のリクエストを検知し，アラートを上げる機能を備えている			✓	2.0
9.9	クライアントの記憶域 (HTML5 のローカルストレージ，セッションストレージ，IndexedDB，通常の Cookie，Flash Cookie など) に保存されるデータにセンシティブなデータや PII が含まれていない	✓	✓	✓	3.0
9.10	センシティブなデータの収集がデータ保護条令に基づいて行われる，あるいはアクセスログの取得が必須である場合，センシティブなデータに対するアクセスがログに記録される		✓	✓	3.0
9.11	センシティブなデータが不要になるや直ちにメモリから消去され，フレームワークやライブラリ，オペレーティングシステムがサポートする機能や手法に従って処理される		✓	✓	3.0

参考情報

詳しくは以下の情報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: 通信のセキュリティに関する検証要件

管理目標

検証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.

- センシティブなデータの送信には TLS を使用する
- 常に協力なアルゴリズムと暗号を使用する

要件

#	説明	1	2	3	導入
10.1	信頼できる認証局(CA) から TLS サーバー証明書までのパスを構築することができ，各サーバー証明書が有効である	✓	✓	✓	1.0
10.3	認証済みあるいはセンシティブなデータや機能をやりとりする通信経路（外部通信およびバックエンド方式による通信を含む）がすべて TLS で暗号化されており，安全性の低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ない．最も有力な代替手段に推奨アルゴリズムを使っている.	✓	✓	✓	3.0
10.4	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			✓	1.0
10.5	すべてのクライアント証明書について，トラストアンカーと失効情報を用いて認証パスの構築と確認が行われる			✓	1.0
10.6	センシティブな情報や機能を持つ外部システムとの接続がすべて認証されている		✓	✓	1.0
10.8	アプリケーションで単一の標準的 TLS 実装を使用する．また承認された動作モードで機能するよう設定されている.			✓	1.0
10.10	TLS 証明書の公開鍵ピニングが，実稼働用とバックアップ用の両方の公開を使ってで実装されている．詳しくは参考情報を参照.			✓	3.0
10.11	すべてのリクエストのすべてのサブドメインについて，Strict-Transport-Security: max-age=15724800; includeSubdomains のような HTTP Strict Transport Security ヘッダが含まれる	✓	✓	✓	3.0

#	説明	1	2	3	導入
10.12	実運用中の Web サイト の URL が，Web ブラウザベンダが管理する Strict Transport Security preload list に送信されている．詳しくは参考情報を参照．			✓	3.0
10.13	トラフィックが受動的に記録される攻撃のリスクを軽減するために，Forward Secrecy を実現できる暗号を使用している	✓	✓	✓	3.0
V10.14	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．	✓	✓	✓	3.0
V10.15	すべての証明書階層において（独自に使用する認証局も含まれる），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
V10.16	TLS に関する設定が，現時点における最善の実践と合っている（一般的な構成，暗号，アルゴリズムは安全でなくなるため）	✓	✓	✓	3.0

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.**
  https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
- **“認定された 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困難であったり，矛盾が生じたり，また混乱を招く可能性があるでしょう．要件 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)などのガイドを参照し，既知の適切な構成をつくり (https://mozilla.github.io/server-side-tls/ssl-config-generator/) sslyze などの既存の TSL 評価ツールや脆弱性スキャナ，信頼できるオンラインの TLS アセスメントサービスを使用して，望ましいレベルのセキュリティを確保することです．本セクションに準拠しない例としては，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一般に見られます．

- **証明書ピニング**. 詳しくは https://tools.ietf.org/html/rfc7469 を参照してください．実運用中およびバックアップ用の鍵に証明書ピニングを行う根拠はビジネスの継続性にあります． https://noncombatant.org/2015/05/01/about-http-public-key-pinning/ を参照してください．
- **Pre-loading HTTP Strict Transport Security**
  https://www.chromium.org/hsts

V11: HTTP のセキュリティ設定に関する検証要件

管理目標

検証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がデフォルトの構成よりも適切に強化されている
- HTTP レスポンスの Content-Type ヘッダが安全な文字セットで構成されている

要件

#	説明	1	2	3	導入
11.1	アプリケーションが定義済みの HTTP リクエストメソッド（GET と POST など）のみを受け付け，使用しないメソッド (TRACE，PUT，DELETE など) を明示的に拒否する	✓	✓	✓	1.0
11.2	HTTP レスポンスに安全な文字セット (UTF-8 や ISO 8859-1) を指定するコンテンツタイプヘッダが含まれる	✓	✓	✓	1.0
11.3	信頼されるプロキシや SSO デバイスが追加する HTTP ヘッダー（ベアラトークンなど）がアプリケーションによって認証される		✓	✓	2.0
11.4	第三者の X-Frame へのコンテンツ表示を許可しないサイトで Content Security Policy V2 (CSP) を使用する		✓	✓	2.0
11.5	HTTP ヘッダや HTTP レスポンスを通じてシステムのコンポーネントの詳細なバージョン情報が開示されない	✓	✓	✓	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"` (コンテントタイプと適切なファイル名) が含まれている	✓	✓	✓	3.0
11.7	Content Security Policy V2 (CSP) を使ってインライン JavaScript を無効にしている．あるいは CSP nonce またはハッシュを使ってインライン JavaScript に対する完全性を確認している	✓	✓	✓	3.0
11.8	次のヘッダが存在する． `X-XSS-Protection: 1; mode=block`	✓	✓	✓	3.0

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 HTTP Verb Tampering
  https://www.owasp.org/index.php/Testing_for_HTTP_Verb_Tampering_%28OTG-INPVAL-003%29
- API のレスポンスに `Content-Disposition` ヘッダを追加することでクライアントとサーバ間での MIME タイプの誤解釈を悪用する多くの攻撃を防止することができます. また **“`filename`”** オプションの追加は Reflected File Download 攻撃に対する対策として有効です.
  https://www.blackhat.com/docs/eu-14/materials/eu-14-Hafif-Reflected-File-Download-A-New-Web-Attack-Vector.pdf

V12: セキュリティ設計に関する検証要件

本セクションは Application Security Verification Standard 2.0 の要件 V11 に取り込まれました.

V13: 悪性活動の管理に関する検証要件

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，ロジック上の欠陥が作り込まれていない

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アをもってしても，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.

要件

#	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	コードレビューにより，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

参考情報

詳しくは以下の情報を参照してください.

- http://www.dwheeler.com/essays/apple-goto-fail.html

V14: 内部セキュリティに関する検証要件

本セクションは Application Security Verification Standard 2.0 の要件 V13 に取り込まれました.

V15: ビジネスロジックに関する検証要件

管理目標

検証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.

- ビジネスロジックが正しい順序で処理されている
- 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いる. また, なりしまし (spoofing), 改ざん (tampering), 否認 (repudiation), 情報の漏洩 (information disclosure), 権限昇格 (elevation of privilege) 攻撃の対策を行っている.

要件

#	説明	1	2	3	導入
V15.1	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, 全ステップが人間が実際に処理する時間で処理され, 誤った順序での処理やステップの省略, 別ユーザのステップの処理, 送信されるのが早すぎるトランザクションの処理がいずれも行われない		✓	✓	2.0
V15.2	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, ユーザ毎にそれが適用されている. 自動もしくは異常な攻撃に対して警告を上げたり自動応答する設定を行える		✓	✓	2.0

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: 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証要件

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，webroot の外に保存され，アクセスが制限される

要件

#	説明	1	2	3	導入
16.1	URL のリダイレクトおよび転送において，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た信頼できないファイルのデータがファイル入出力コマンドで直接使用されない (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 タイプ，OS コマンドインジェクションの脆弱性を防止するため)	✓	✓	✓	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	✓	✓	✓	2.0
16.4	リモートもしくは ローカルファイルインクルードの脆弱性対策のため，信頼できないデータはインクルード，クラスローダー，リフレクションに使用されない	✓	✓	✓	2.0
16.5	遠隔の任意のコンテンツが混入しないよう，信頼できないデータをクロスドメインリソース共有 (CORS) で使用しない	✓	✓	✓	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 の外にパーミッションを制限した上で保存され，できれば強力な検証が行われている		✓	✓	3.0
16.7	Web サーバもしくはアプリケーションサーバーがデフォルトで遠隔のリソースやシステムへのアクセスを拒否するように設定されている		✓	✓	2.0

#	説明	1	2	3	導入
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 など 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:
  https://www.owasp.org/index.php/Unrestricted_File_Upload

V17: モバイルに関する検証要件

管理目標

本セクションではモバイルアプリに固有の管理策を扱います．バージョン 2.0 以降，セクション間で重複する管理策は削除されているため，関連する他のセクションと合わせて考慮する必要があります．

モバイルアプリが次の要件を満たすことを検証します．

- サーバにおける管理策と同レベルのセキュリティ管理策をモバイルクライアントが備える
- デバイス上に保存されるセンシティブな情報資産がセキュアな方法で保存されている
- デバイスからのセンシティブなデータの送信はトランスポート層のセキュリティを考慮して行われる

要件

#	説明	1	2	3	導入
17.1	UDID や IMEI 番号など，デバイス上に保存されかつ他のアプリが取得可能な ID 値を認証トークンとして使用しない	✓	✓	✓	2.0
17.2	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) にアプリが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の中であっても)	✓	✓	✓	2.0
17.4	秘密鍵，API トークン，パスワードがアプリ内で動的に生成される		✓	✓	2.0
17.5	アプリがセンシティブな情報の漏えいを防止している．情報えいの例としては，実行中のアプリのスクリーンショットがバックグラウンドに切り替わる際に保存される，センシティブな情報をコンソールに書き出す，などがある．		✓	✓	2.0
17.6	必要な機能とリソースに対する最小限のアクセス許可をアプリが要求する		✓	✓	2.0

#	説明	1	2	3	導入
17.7	アプリのセンシティブなコードが予測できない形でメモリ上に配置される (ASLR など)	✓	✓	✓	2.0
17.8	モバイルアプリに対するデバッガ (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を導入している			✓	2.0
17.9	センシティブな Activity, Intent, Content Provider 等を同一デバイス上の他のアプリに export していない	✓	✓	✓	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し，不要時に上書きされる (メモリ分析攻撃による被害の軽減)			✓	2.0
17.11	アプリが公開する Activity, Intent, Content Provider においてすべての入力を検証する	✓	✓	✓	2.0

参考情報

詳しくは以下の情報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証要件

管理目標

RESTful または SOAP ベースの Web サービスを使用する検証対象のアプリケーションが，次の機能を備えることを確認します．

- すべての Web サービスについて，適切な認証とセッション管理，認可
- 低信頼レベルから高信頼レベルへと遷移するすべてのパラメーターの入力値検証
- API を提供する SOAP Web サービスレイヤーの基本的な相互運用性

要件

#	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 サービス管理者のみが Web アプリケーション内の管理機能にアクセスできる	✓	✓	✓	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を行っている．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" および類似のものを使用しない	✓	✓	✓	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 が期待するもの (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のデータ交換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

#	説明	1	2	3	導入
18.10	代替のセキュリティレベルの低いアクセスパスが存在しない		✓	✓	3.0

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: Configuration and Deployment Management Testing
  https://www.owasp.org/index.php/Testing_for_configuration_management

V19. 構成

管理目標

検証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.

- ライブラリとプラットフォームが最新である
- デフォルトの設定がセキュアである
- ユーザが設定をデフォルトに変更してもシステムの基盤にセキュリティ上の弱点や欠陥が生じないよう，十分ハードニングされている

要件

#	説明	1	2	3	導入
19.1	すべての構成要素が，最新にアップデートされ，適切なセキュリティ設定とバージョンになっている．つまり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など）が削除されている	✓	✓	✓	3.0
19.2	コンポーネント間の通信（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）が暗号化されている（特に，各コンポーネントが異なるコンテナーやシステム上にまたがる場合）		✓	✓	3.0
19.3	コンポーネント間の通信（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）の前に，必要最小限の権限を持つアカウントを使って認証が行われる．		✓	✓	3.0
19.4	適切なサンドボックス化やコンテナー化，あるいは隔離を行った上でアプリケーションが配備されており，攻撃者による他のアプリケーションに対する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のプロセスがセキュアである		✓	✓	3.0

#	説明	1	2	3	導入
19.6	認可された管理者がセキュリティに関連するすべての設定の完全性を検査できる機能を持ち，設定が改ざんされていないことを確認している			✓	3.0
19.7	アプリケーションのすべてのコンポーネントが署名されている			✓	3.0
19.8	信頼できるリポジトリから取得したサードパーティのコンポーネントを使用している			✓	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: 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

Original #	Description	Status	Removed	Reason
2.3	Verify that if a maximum number of authentication attempts is exceeded, the account is locked for a period of time long enough to deter brute force attacks.	Deprecated	2.0	A more complex requirement replaced it (v2.20)
2.5	Verify that all authentication controls (including libraries that call external authentication services) have a centralized implementation.	Merged	3.0	Genericized to include all security controls and moved to 1.10
2.10	Verify that re-authentication is required before any application- specific sensitive operations are permitted.	Deprecated	2.0	Re-authentication is so rarely observed that we decided to remove the control
2.11	Verify that after an administratively-configurable period of time, authentication credentials expire.	Deprecated	2.0	Absolute timeouts and credential expiry removed as not being an effective control.
2.14	Verify that all authentication credentials for accessing services external to the application are encrypted and stored in a protected location (not in source code).	Updated	2.0	Became V2.21
2.15	Verify that all code implementing or using authentication controls is not affected by any malicious code.	Moved	2.0	Moved to V13 - Malicious Code
3.8	Verify that the session id is changed upon re-authentication	Updated	3.0	Rolled into 3.7
3.9	Verify that the session id is changed or cleared on logout	Updated	3.0	Rolled into 3.7
3.13	Verify that all code implementing or using session management controls is not affected by any malicious code	Moved	2.0	Moved to V13 - Malicious code

3.14	Verify that authenticated session tokens using cookies are protected by the use of "HttpOnly".	Updated	3.0	Moved into 3.13
3.15	Verify that authenticated session tokens using cookies are protected with the "secure" attribute.	Updated	3.0	Moved into 3.13
4.2	Verify that users can only access secured URLs for which they possess specific authorization.	Updated	3.0	Rolled into 4.1
4.3	Verify that users can only access secured data files for which they possess specific authorization.	Updated	3.0	Rolled into 4.1
4.13	Verify that limitations on input and access imposed by the business on the application (such as daily transaction limits or sequencing of tasks) cannot be bypassed.	Moved	3.0	Moved to V15 Business Logic
4.15	Verify that all code implementing or using access controls is not affected by any malicious code.	Moved	2.0	Moved to V13 Malicious Controls
5.2	Verify that a positive validation pattern is defined and applied to all input	Deprecated	2.0	Removed as too difficult to implement particularly for free form text inputs
5.4	Verify that a character set, such as UTF-8, is specified for all sources of input	Deprecated	3.0	Removed as too difficult to implement in most languages
5.7	Verify that all input validation failures are logged.	Deprecated	3.0	Removed as would create too many useless logs that would be ignored
5.8	Verify that all input data is canonicalized for all downstream decoders or interpreters prior to validation.	Deprecated	3.0	Removed as Type 1 JSP technology specific and not an issue for most modern frameworks
5.9	Verify that all input validation controls are not affected by any malicious code	Moved	2.0	Moved to V13 Malicious controls

5.14	Verify that the runtime environment is not susceptible to XML Injections or that security controls prevents XML Injections	Merged	3.0	Merged with V5.13
5.15	-- EMPTY REQUIREMENT --	Deleted	3.0	This requirement never existed
5.19	Verify that for each type of output encoding/escaping performed by the application, there is a single security control for that type of output for the intended destination	Merged	3.0	Genericized to include all security controls and moved to 1.10
7.1	Verify that all cryptographic functions used to protect secrets from the application user are implemented server side	Deprecated	3.0	Many modern responsive and mobile apps include this by design
7.3	Verify that access to any master secret(s) is protected from unauthorized access (A master secret is an application credential stored as plaintext on disk that is used to protect access to security configuration information).	Moved	3.0	Moved to V2.29
7.4	Verify that password hashes are salted when they are created	Moved	2.0	Moved to V2.13
7.5	Verify that cryptographic module failures are logged	Deprecated	2.0	Creating unnecessary logs that are never reviewed is counterproductive
7.10	Verify that all code supporting or using a cryptographic module is not affected by any malicious code	Moved	2.0	Moved to V13
8.2	Verify that all error handling is performed on trusted devices		3.0	Deprecated
8.3	Verify that all logging controls are implemented on the server.	Moved	3.0	Became a more generic architectural control V1.13
8.9	Verify that there is a single application-level logging implementation that is used by the software.	Moved	3.0	Became a more generic architectural control V1.13

8.11	Verify that a log analysis tool is available which allows the analyst to search for log events based on combinations of search criteria across all fields in the log record format supported by this system.	Deprecated	3.0	Removed as not required for secure software
8.12	Verify that all code implementing or using error handling and logging controls is not affected by any malicious code.	Moved	2.0	Moved to V13 Malicious Controls
8.15	Verify that logging is performed before executing the transaction. If logging was unsuccessful (e.g. disk full, insufficient permissions) the application fails safe. This is for when integrity and non-repudiation are a must.	Deprecated	3.0	Removed as too detailed a control that would only be applicable to small percentage of all apps
10.2	Verify that failed TLS connections do not fall back to an insecure HTTP connection	Merged	3.0	Merged with 10.3
10.7	Verify that all connections to external systems that involve sensitive information or functions use an account that has been set up to have the minimum privileges necessary for the application to function properly			
10.9	Verify that specific character encodings are defined for all connections (e.g., UTF-8).			
V11.1	Deprecated			
V11.4	Deprecated			
V11.5	Deprecated			
V11.6	Deprecated			
V11.7	Deprecated			
V11.8	Deprecated			
V11.4	Deprecated			
V13.1	Deprecated			

V13.2	Deprecated			
V13.3	Deprecated			
V13.4	Deprecated			
V13.5	Deprecated			
V13.6	Deprecated			
V13.7	Deprecated			
V13.8	Deprecated			
V13.9	Deprecated			
15.1-15.7 15.9	Business Logic Section.	Merged	3.0	Most of section 15 has been merged into 15.8 and 15.10.
15.11	Verify that the application covers off risks associated with Spoofing, Tampering, Repudiation, Information Disclosure, and Elevation of privilege (STRIDE).	Duplicate	3.0	Duplicated requirement. Captured by V1.6
16.4	Verify that parameters obtained from untrusted sources are not used in manipulating filenames, pathnames or any file system object without first being canonicalized and input validated to prevent local file inclusion attacks.	Moved	3.0	Moved to V16.2
17.1	Verify that the client validates SSL certificates	Deprecated	3.0	Duplicated requirement. General requirement already captured by V10.
V17.7	Deprecated			
V17.8	Deprecated			
V17.10	Deprecated			
V17.11	Deprecated			

V17.12	Deprecated			
V17.13	Deprecated			
V17.14	Deprecated			
V17.15	Deprecated			
V17.16	Deprecated			
V17.17	Deprecated			
V17.18	Deprecated			
V17.19	Deprecated			
V17.20	Deprecated			
V17.22	Deprecated			
V17.23	Deprecated			
V17.24	Deprecated			

付録 B: 用語集

- **アクセス制御 (Access Control)**：ファイルや機能，URL，データに対するアクセスを，ユーザが所属するグループや ID に基づいて制限する手段.
- **Address Space Layout Randomization (ASLR)**：バッファオーバーフロー攻撃対策の一手法.
- **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，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.
- **バックドア (Back Doors)**：アプリケーションに対して許可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や操作のリスト(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)：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 .
- **認証局** (CA)：電子証明書を発行する機関.
- **通信のセキュリティ (Communication Security)**：アプリケーションのコンポーネント間，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の間で交換されるデータの保護.
- **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つ.

- **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.
- **サービス運用妨害攻撃 (Denial of Service (DoS) Attacks)**：アプリケーションが処理できる限界を超えたリクエストを送信する攻撃.
- **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.
- **イースターエッグ (Easter Eggs)**：悪性コードの一種．ある特定の入力イベントが発生す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，サーバサイドアプリケーションやサービス.
- **FIPS 140-2**：暗号モジュールの設計と実装の検証に使用される標準.
- **グローバルー意識別子 (Globally Unique Identifier, GUID)**：ソフトウェアにおいて識別子として使用されるユニークな照会番号.
- **ハイパーテキストマークアップ言語 (HyperText Markup Language, HTML)**：Web ブラウザに表示される Web ページや他の情報の作成に主として用いられるマークアップ言語.
- **ハイパーテキスト転送言語 (Hyper Text Transfer Protocol, HTTP)**：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.
- **入力検証 (Input Validation)**： 信頼できないユーザ入力を正規化および検証する.
- **Lightweight Directory Access Protocol (LDAP)**：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と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を回避する．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の向上に注力する，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ある．http://www.owasp.org/ を参照.
- **出力のエンコード (Output encoding)**：Web ブラウザや外部システムに渡す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 (Personally Identifiable Information, PII)**：単独でもしくは他の情報と合わせて使用することで，個人の識別，接触，位置の特定，あるいはコンテキストにおける個人の識別を可能にする情報.
- **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)**：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する.
- **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(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(監査レコードの生成など)，機能や構成要素.
- **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る.
- **静的検証 (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出する自動化ツールを使った検証.

- **検証対象 (Target of Verification, TOV)**：OWASP ASVS の要件に則ってアプリケーションのセキュリティ検証を行う際に，検査対象となる特定のアプリケーション．このような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.
- **トランスポート層のセキュリティ (Transport Layer Security)：**インターネットの通信セキュリティを提供する暗号プロトコル.
- **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．多くの場合リソースへの参照として使用される.
- **ユーザ受入れテスト (User acceptance testing, UAT)**：実稼働環境と同じように動作するテストのための環境．ソフトウェアを実稼働する前に，あらゆるテストがこの環境で実施する.
- **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.
- **ホワイトリスト (Whitelist)**：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 (入力検証で許可されている文字のリストなど).
- **XML**：ドキュメントの符号化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: 参考情報

The following OWASP projects are most likely to be useful to users/adopters of this standard:

- 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

Similarly, the following web sites are most likely to be useful to users/adopters of this standard:

- 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: 他の標準との対応

PCI DSS 6.5 is derived from the OWASP Top 10 2004/2007, with some recent process extensions. The ASVS is a strict superset of the OWASP Top 10 2013 (154 items to 10 items), so all of the issues covered by OWASP Top 10 and PCI DSS 6.5.x are handled by more fine grained ASVS control requirements. For example, “Broken authentication and session management” maps exactly to sections V2 Authentication and V3 Session Management.

Full mapping is achieved by verification level 3, although verification level 2 will address most PCI DSS 6.5 requirements except 6.5.3 and 6.5.4. Process issues, such as PCI DSS 6.5.6, are not covered by the ASVS.

PCI-DSS 3.0	ASVS 3.0	Description
6.5.1 Injection flaws, particularly SQL injection. Also consider OS Command Injection, LDAP and XPath injection flaws as well as other injection flaws	5.11, 5.12, 5.13, 8.14, 16.2	Exact mapping.
6.5.2 Buffer overflows	5.1	Exact mapping
6.5.3 Insecure cryptographic storage	v7 - all	Comprehensive mapping from Level 1 up
6.5.4 Insecure communications	v10 - all	Comprehensive mapping from Level 1 up
6.5.5 Improper error handling	3.6, 7.2, 8.1, 8.2	Exact mapping
6.5.7 Cross-site scripting (XSS)	5.16, 5.20, 5.21, 5.24, 5.25, 5.26, 5.27, 11.4,11.15	ASVS breaks down XSS into several requirements highlighting the complexity of XSS defense especially for legacy applications
6.5.8 Improper Access Control (such as insecure direct object references, failure to restrict URL access, directory traversal and failure to restrict user access to functions).	v4 - all	Comprehensive mapping from Level 1 up

6.5.9 Cross-site request forgery (CSRF).	4.13	Exact mapping. ASVS considers CSRF defense to be an aspect of access control.
6.5.10 Broken authentication and session management.	v2 and v3 - all	Comprehensive mapping from Level 1 up