



OWASP
Open Web Application
Security Project

Standard

Mobile AppSec Verification

Version 1.1

(Japanese Translation)

Project leaders: Sven Schleier and Jeroen Willemsen

Creative Commons (CC) Attribution Share-Alike

Free version at <http://www.owasp.org>



目次

変更履歴	1.1
序文	1.2
扉	1.3
モバイルアプリケーションセキュリティ検証標準	1.4
監査と認定	1.5

セキュリティ要件

V1: アーキテクチャ、設計、脅威モデリング要件	2.1
V2: データストレージとプライバシー要件	2.2
V3: 暗号化要件	2.3
V4: 認証とセッション管理要件	2.4
V5: ネットワーク通信要件	2.5
V6: プラットフォーム相互作用要件	2.6
V7: コード品質とビルド設定要件	2.7
V8: 耐性要件	2.8

付録

付録 A: 用語集	3.1
付録 B: 参考情報	3.2

Changelog

This document is automatically generated at Sat Dec 29 2018 08:58:56 GMT+0100 (Midden-Europese standaardtijd)

WIP - 1.1.2 Sponsorship and internationalization

- Added thank you note for buyers of the book.
- Added missing authentication link & updated broken authentication link in V4.
- Fixed swap of 4.7 and 4.8 in english.
- First international release!
- Fixes in Spanish translation. Translation is now in sync with English (1.1.2).
- Fixes in Russian translation. Translation is now in sync with English (1.1.2).
- Added first release of German, French and Japanese!
- Simplified document for ease of translation.
- Added instructions for automated releases.

1.1.0 14 July 2018:

The following changes are part of release 1.1:

- Requirement 2.6 "The clipboard is deactivated on text fields that may contain sensitive data." was removed.
- Requirement 2.2 "No sensitive data should be stored outside of the app container or system credential storage facilities." was added.
- Requirement 2.1 was reworded to "System credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys.".

1.0 12 January 2018:

The following changes are part of release 1.0:

- Delete 8.9 as the same as 8.12
- Made 4.6 more generic
- Minor fixes (typos etc.)

序文

OWASP モバイルプロジェクト Bernhard Mueller による序文。技術革新は迅速に起こります。一昔前、スマートフォンは小さいキーボードを持つ魅力のない端末、技術に精通したビジネスユーザーのための高価な玩具でした。今日、スマートフォンは私たちの生活に不可欠なものです。私たちは情報、ナビゲーション、コミュニケーションのためにそれらに頼っています。ビジネスや社会生活の中の至るところにあります。

すべての新しいテクノロジーは新しいセキュリティリスクをもたらし、その変化に対応することはセキュリティ業界が直面する主要な課題のひとつです。防御側は常に数ステップ遅れます。例えば、多くのデフォルトの対応は物事をなすための古い方法を適用することでした。スマートフォンは小さなコンピュータと似ており、モバイルアプリはクラシックなソフトウェアと同様ですから、確かにセキュリティ要件は似ているでしょう？しかしそのようにはいきません。スマートフォンのオペレーティングシステムはデスクトップオペレーティングシステムとは異なりますし、モバイルアプリはウェブアプリとは異なります。例えば、従来の手法であるシグネチャベースのウイルススキャンは最新のモバイルOS環境では意味を成しません。モバイルアプリの配信モデルと互換性がないだけでなく、サンドボックスの制限により技術的にも不可能です。また、バッファオーバーフローやXSSの問題などいくつかの脆弱性クラスは、デスクトップアプリやWebアプリケーションよりも（例外はありますか）一般的なモバイルアプリのコンテキストではありません。

時間の経過と共に、私たちの業界はモバイル脅威の状況を把握してきました。結局のところ、モバイルセキュリティはデータ保護に関するすべてのことです。アプリは私たちの個人情報、写真、録音、メモ、アカウントデータ、ビジネス情報、位置などを保存します。アプリは私たちが毎日使っているサービスに接続するクライアントとして動作しており、それぞれすべてのメッセージを処理するコミュニケーションハブとして他の人とやり取りします。人のスマートフォンに侵入し、その人の人生にアクセスすることができます。モバイルデバイスがとてもたやすく失くしたり盗まれたりし、モバイルマルウェアが増加していることを考へるとき、データ保護の必要性がさらに明らかになります。

したがってモバイルアプリのセキュリティ標準はモバイルアプリが機密情報を処理、保存、保護する方法に焦点を当てる必要があります。iOSやAndroidなどの最新のモバイルオペレーティングシステムでは安全なデータストレージと通信のための優れたAPIを提供していますが、効果を発揮するには正しく実装され使用されている必要があります。データストレージ、アプリ間通信、暗号APIの適切な使い方や安全なネットワーク通信は慎重な検討が必要な局面のほんの一部です。

データの機密性と完全性を保護するためにどれだけ正しく行うべきかが業界のコンセンサスを必要とする重要な質問です。例えば、私たちの多くはモバイルアプリがTLSエクスチェンジの中でサーバー証明書を検証する必要があることに同意します。しかしSSL証明書ピンニングはどうでしょう？それを行わないと脆弱性が生じるでしょうか？アプリが機密データを処理する場合にそれを要件とすべきでしょうか？それとも逆効果でしょうか？OSがアプリをサンドボックス化していても、SQLiteデータベースに格納されるデータを暗号化する必要があるでしょうか？あるアプリに適切なものでも別のアプリには非現実的かもしれません。MASVSはさまざまな脅威シナリオにあう検証レベルを使用してこれらの要件を標準化する試みです。

さらに、ルートマルウェアやリモート管理ツールの登場によりモバイルオペレーティングシステム自体に悪用可能な欠陥が存在することが認識されています。そのため、機密性の高いデータへの追加保護を提供し、クライアント側の改竄を防止するために、コンテナ化戦略がますます使用されています。物事は複雑に入り組んでいます。ハードウェア支援のセキュリティ機能やAndroid for WorkやSamsung KnoxなどのOSレベルのコンテナ化ソリューションが存在しますが、さまざまなデバイスの間で一貫して利用できるわけではありません。バンドエイドとして、ソフトウェアベースの保護対策を実装できますが、残念ながら、このような種類の保護を検証するための標準やテストプロセスはありません。

結果として、モバイルアプリセキュリティテストレポートは混沌としています。例えば、あるテスト技術者はセキュリティ上の欠陥としてAndroidアプリでの難読化やルート検出の欠如を報告しています。一方、文字列の暗号化、デバッガの検出、制御フローの難読化などの処置は必須ではありません。しかし、このような二元的な方法は理にかなっ

ていません。耐性は二元的な命題ではないためです。これは守ることを目指す特定のクライアント側の脅威に依存します。ソフトウェア保護は無駄ではありませんが、最終的にバイパスすることが可能であるため、ソフトウェア保護をセキュリティコントロールの代わりに使用してはいけないということです。

MASVSの全体的な目標はモバイルアプリケーションセキュリティのベースライン (MASVS-L1) を提供することであり、多層防御処置 (MASVS-L2) やクライアント側の脅威に対する保護 (MASVS-R) も可能にします。MASVSは以下を達成するためのものです。

- セキュアなモバイルアプリケーションを開発しようとするソフトウェアアーキテクトや開発者に要件を提供する
- モバイルアプリのセキュリティコードレビューとしてテストできる業界標準を提供する
- モバイルセキュリティにおけるソフトウェア保護メカニズムの役割を明確にして、その有効性を検証するための要件を提供する
- さまざまなユースケースに対してどのレベルのセキュリティが推奨されるかに関する具体的な推奨を提供する

私たちは100%の業界のコンセンサスが達成不可能であることを認識しています。それでも、私たちはMASVSがモバイルアプリの開発やテストのすべてのフェーズでガイダンスを提供するのに役立つことを願っています。オープンソース標準として、MASVSは時間の経過と共に進化するでしょう。そして、私たちは寄稿や提案を歓迎します。

扉

本標準について

Mobile Application Security Verification Standard (MASVS) 1.1 へようこそ。MASVS は iOS および Android 上のセキュアなモバイル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のフレームワークを確立するためのコミュニティの取組みです。

MASVS 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。私たちは本標準が時間の経過とともに進化することを期待しており、コミュニティからのフィードバックを歓迎します。私たちと連絡を取る最善の方法は OWASP Mobile Project Slack チャンネルを利用することです。

https://owasp.slack.com/messages/project-mobile_omtg/details/

アカウントは以下の URL で作成できます。

<http://owasp.herokuapp.com/>

著作権とライセンス



Copyright © 2018 The OWASP Foundation. 本著作物は [Creative Commons Attribution-ShareAlike 4.0 International License](#) に基づいてライセンスされています。再使用または配布する場合は、他者に対し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。

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
Sven Schleier & Jeroen Willemsen	Bernhard Mueller	Alexander Antukh, Mesheryakov Aleksey, Bachevsky Artem, Jeroen Beckers, Vladislav Chelnokov, Ben Cheney, Stephen Corbiaux, Manuel Delgado, Ratchenko Denis, Ryan Dewhurst, Tereshin Dmitry, Christian Dong, Oprya Egor, Ben Gardiner, Rocco Gränitz, Henry Hu, Sjoerd Langkemper, Vinícius Henrique Marangoni, Martin Marsicano, Roberto Martelloni, Gall Maxim, Rio Okada, Abhinav Sejpal, Stefaan Seys, Yogesh Sharmra, Prabhant Singh, Nikhil Soni, Anant Shrivastava, Francesco Stillavato, Romuald SZKUDLAREK, Abdessamad Temmar, Koki Takeyama, Chelnokov Vladislav

本ドキュメントは Jim Manico により執筆された OWASP Web アプリケーションセキュリティ検証標準のフォークとして始まりました。

スポンサー

MASVS と MSTG のいずれもコミュニティにより無償奉仕で作成および維持されていますが、時にはいくつかの外的支援が必要となることもあります。したがって、テクニカルエディタを雇うことができる資金を提供したスポンサーに感謝します。彼らのスポンサーシップは MASVS や MSTG の内容にいかなる形であれ影響を与えないことに注意します。スポンサーシップパッケージは [OWASP Project Wiki](#) に記載されています。

名誉後援者



次に、OWASPベイエリア支部の後援に感謝します。最後に、Leanpubからその本を買ってそのように私たちに後援してくれたみんなに感謝したいです。

モバイルアプリケーションセキュリティ検証標準

MASVSを使用することでモバイルアプリのセキュリティに信頼レベルを確立することができます。要件は以下の目標を念頭において開発されました。

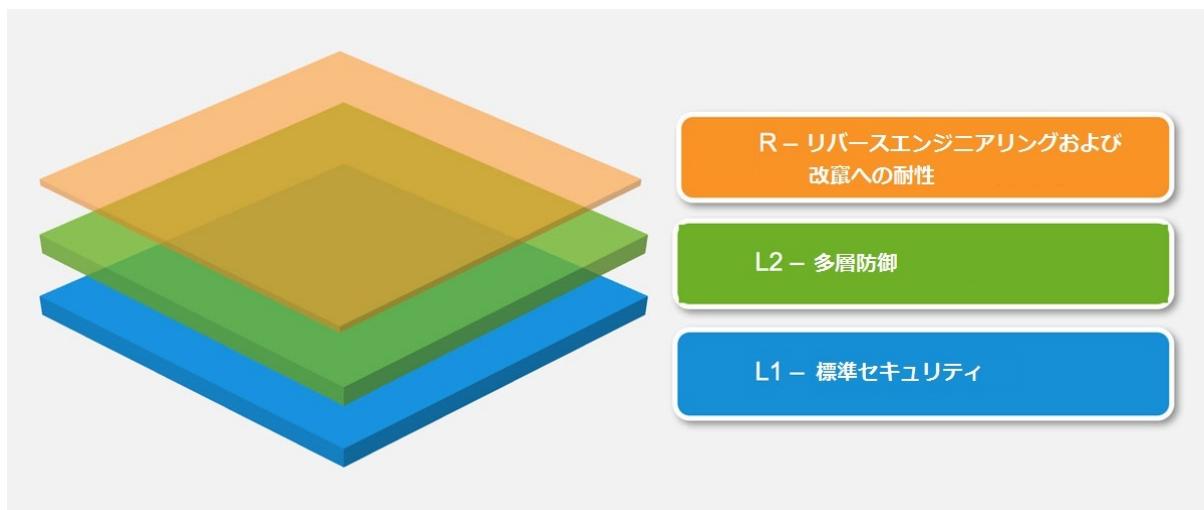
- メトリックとして使用する - 開発者およびアプリケーション所有者により既存のモバイルアプリと比較できるセキュリティ標準を提供すること。
- ガイダンスとして使用する - モバイルアプリ開発およびテストのすべてのフェーズでのガイダンスを提供すること。
- 調達時に使用する - モバイルアプリのセキュリティ検証のベースラインを提供すること。

モバイルアプリセキュリティモデル

MASVSは2つの厳格なセキュリティ検証レベル(L1およびL2)とフレキシブルである一連のリバースエンジニアリング耐性要件(MASVS-R)を定義しています。MASVS-L1 および MASVS-L2 は一般的なセキュリティ要件であり、すべてのモバイルアプリ(L1)および機密性の高いデータを扱うアプリ(L2)に推奨されます。MASVS-R は追加の保護コントロールを対象としています。クライアント側の脅威を防止することが設計の目標である場合に適用できます。

MASVS-L1 の要件を満たすことで、セキュリティベストプラクティスに従い、一般的な脆弱性に苦しむことのないセキュアなアプリケーションを実現します。MASVS-L2 は SSL ピンニングなどの追加の多層防御コントロールを加えて、より洗練された攻撃に対し耐性のあるアプリをもたらします。モバイルオペレーティングシステムのセキュリティコントロールが損なわれていないこと、およびエンドユーザーが潜在的な敵対者とみなされないことを仮定しています。MASVS-R のソフトウェア保護要件のすべてもしくは一部を満たすことは、エンドユーザーに悪意のある場合やモバイルOSが侵害された場合に、特定のクライアントサイドの脅威に対しての防御に役立ちます。

注意。MASVS-R や OWASP モバイルテストガイドに記載されているソフトウェア保護コンロトールは究極的にはバイパスでき、セキュリティコントロールの代わりに使用してはいけません。代わりに、MASVS L1 もしくは L2 の MASVS 要件を満たすアプリに、脅威に特有の追加の保護コントロールを加えることを意図しています。



ドキュメントの構成

MASVS の最初のパートには、セキュリティモデルおよび利用可能な検証レベルの説明と、実際に標準の使用方法に関する推奨事項が含まれます。次のパートには、詳細なセキュリティ要件と検証レベルへのマッピングが記載されています。要件は技術的な目的 / スコープに基づいて8つのカテゴリ (V1~V8) に分類されています。MASVS および MSTG 全体で以下の命名法が使用されています。

- 要件カテゴリ: MASVS-Vx, 例. MASVS-V2: データストレージとプライバシー
- 要件: MASVS-Vx.y, 例. MASVS-V2.2: 「機密データがアプリケーションログに書き込まれていない。」

検証レベルの詳細

MASVS-L1: 標準セキュリティ

MASVS-L1 を実現するモバイルアプリはモバイルアプリケーションセキュリティのベストプラクティスに準拠しています。コード品質、機密データの取り扱い、モバイル環境との連携に関して基本的な要件を満たしています。セキュリティコントロールを検証するにはテストプロセスを実行する必要があります。このレベルはすべてのモバイルアプリケーションに適しています。

MASVS-L2: 多層防御

MASVS-L2 は標準要件を超える高度なセキュリティコントロールを導入します。L2 を満たすためには、脅威モデルが存在する必要があり、セキュリティはアプリのアーキテクチャおよび設計の不可欠な部分である必要があります。このレベルはモバイルバンキングなどの機密データを処理するアプリケーションに適しています。

MASVS-R: リバースエンジニアリングと改竄への耐性

このアプリは最高水準のセキュリティを備えており、機密コードやデータを抽出するための改竄、改造、リバースエンジニアリングなどの特定の明確に定義されたクライアントサイドの攻撃に対しても耐性があります。このようなアプリはハードウェアのセキュリティ機能や十分に強力で検証可能なソフトウェア保護技術を活用しています。MASVS-R は機密性の高いデータを扱うアプリに適用され、知的財産を保護し、アプリの改竄を防止する手段となります。

推奨される使い方

アプリは事前のリスクアセスメントや要求される全体のセキュリティレベルに応じて MASVS L1 もしくは L2 に沿って検証できます。L1 はすべてのモバイルアプリに適用され、L2 は一般的により機密性の高いデータや機能を扱うアプリに推奨されます。MASVS-R (もしくはその一部) は適切なセキュリティ検証に追加することで、再パッケージ化や機密データの抽出などの特定の脅威に対する耐性を検証することができます。

要約すると、以下の検証タイプが利用できます。

- MASVS-L1
- MASVS-L1+R
- MASVS-L2
- MASVS-L2+R

それぞれの組み合わせには異なるグレードのセキュリティと耐性が反映されています。その目的は柔軟性を確保することです。例えば、モバイルゲームにはユーザビリティ上の理由から二要素認証などの MASVS-L2 セキュリティコントロールを加えることは容認されないかもしれません、改竄防止のための強いビジネスニーズがあります。

どの検証タイプを選択するか

MASVS L2 の要件を実装することでセキュリティが向上すると同時に、開発コストが増加し、エンドユーザーエクスペリエンスが悪化する可能性があります（典型的なトレードオフです）。一般的に、L2 はリスク対コストの観点から意味がある場合のアプリに使用すべきです（つまり、妥協した機密性もしくは完全性により引き起こされる潜在的な損失が追加のセキュリティコントロールによりもたらされるコストより高い場合）。リスクアセスメントは MASVS を適用する前の最初のステップであるべきです。

事例

MASVS-L1

- すべてのモバイルアプリ。MASVS-L1 には開発コストとユーザーエクスペリエンスに妥当な影響を与えるセキュリティのベストプラクティスを掲載しています。より高いレベルのものに該当しないアプリには MASVS-L1 の要件を適用します。

MASVS-L2

- ヘルスケア業界：なりすまし犯罪や不正支払などのさまざまな犯罪行為に使用される個人識別情報を保管するモバイルアプリ。米国のヘルスケア分野では、医療保険の相互運用性と説明責任に関する法律 (HIPPA) のプライバシー、セキュリティ、侵害開示に関する規則、患者の安全性に関する規則があります。
- 金融業界：クレジットカード番号、個人情報などの機密性の高い情報にアクセスできたり、ユーザーが送金できるアプリ。これらのアプリは詐欺を防止するための追加のセキュリティコントロールを必要とします。金融アプリはPCIデータセキュリティスタンダード (PCIDSS)、グラム・リーチ・ブライリー法、サーベンス・オクスリー (SOX) 法に準拠する必要があります。

MASVS-L1+R

- IP保護がビジネスゴールであるモバイルアプリ。MASVS-R に掲載されている耐性コントロールは元のソースコードを取得するのに必要な作業を増やし、改竄 / クラッキングを防止するために使用できます。
- ゲーム業界：競争の激しいオンラインゲームなど、改造やチート行為を防止することが必要不可欠なゲーム。チート行為はオンラインゲームでの重要な問題です。大量のチート行為者がプレイヤーの不満を招き、最終的にはゲームが失敗となる可能性があります。MASVS-R はチート行為者への負担を増やすために基本的な耐タンパ性コントロールを提供します。

MASVS-L2+R

- 金融業界：ユーザーが送金できるオンラインバンキングアプリ。侵入したデバイスでのコードインジェクションやインストルメンテーションの技術によるリスクをもたらします。この場合、MASVS-R のコントロールを使用して改竄を防ぐことで、マルウェア作成者のハードルを上げることができます。
- 設計上、モバイルデバイス上に機密データを保存する必要があると同時に幅広いデバイスやオペレーティングシステムのバージョンをサポートしなければならないすべてのモバイルアプリ。この場合、機密データを抽出することを目指す攻撃者の負担を増やすために多層防御策として耐性コントロールを使用することができます。

監査と認定

MASVS 認定と認証マークに対する 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、検証者、ソフトウェアの認定は行っていません。

そのような保証の表明、認証マーク、認定はいずれも OWASP によって公式に検査、登録、認定されたものではありません。組織は (M)ASVS 認定を主張する第三者や認証マークについて、その信頼性を慎重に検討する必要があります。

ただし、OWASP の公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。

モバイルアプリの認定に関するガイダンス

モバイルアプリの MASVS への準拠を検証するために推奨される方法は「オープンブック」レビューを実行することです。つまり、アプリの設計者や開発者、プロジェクト文書、ソースコード、役割ごとに少なくともひとつのユーザーアカウントへのアクセスを含むエンドポイントへの認証されたアクセスといった、主要リソースへのアクセス権限をテスト技術者に割り当てることです。

MASVS は（クライアント側の）モバイルアプリと、アプリとそのリモートエンドポイント間のネットワーク通信のセキュリティを対象とすることに注意することが重要です。また、ユーザー認証とセッション管理に関連するいくつかの基本的かつ一般的な要件も対象とします。アプリに関連付けられた（Webサービスなどの）リモートサービスに対して特定の要件は含まれていません。認証およびセッション管理に関連する限られた一般的な要件に対して安全です。ただし、MASVS V1 ではリモートサービスを全体的な脅威モデルでカバーし、OWASP ASVS などの適切な標準に沿って検証があることを明記しています。

認定機関は、（特に重要な構成要素が範囲外である場合）検証の範囲、合格したテストと不合格のテストを含む検証結果の要約、不合格のテストへの解決法の明確な指示をレポートに含める必要があります。詳細な調書、スクリーンショットやムービー、問題を確実かつ繰り返し利用するためのスクリプト、傍受したプロキシログなどのテストの電子記録、クリーンアップリストなどの関連メモを保持することは標準的な業界の慣行と考えられます。ツールを実行して不合格を報告するだけでは不十分です。認定レベルのすべての問題がテストされ、余すところなくテストされている十分な証跡を提供していません。係争の場合には、すべての検証要件が実際にテストされたことを実証するのに十分な証跡が必要となります。

OWASP モバイルセキュリティテストガイド (MSTG) を使用する

OWASP MSTG はモバイルアプリのセキュリティをテストするためのマニュアルです。MASVS に記載されている要件を検証するための技術的プロセスを記述しています。MSTG には MASVS のそれぞれの要件に対応するテストケースのリストが含まれています。MASVS の要件は高レベルで汎用的ですが、MSTG はモバイル OS ごとに詳細な推奨案とテスト手順を提供しています。

自動セキュリティテストツールの役割

可能な限り効率を上げるためにソースコードスキャナやブラックボックステストツールの使用が奨励されています。ただし、自動ツールのみを使用して MASVS 検証を完了することはできません。モバイルアプリはさまざまであり、全体的なアーキテクチャ、ビジネスロジック、使用されている特定のテクノロジーやフレームワークの技術的な落とし穴を理解することはアプリのセキュリティを検証するために必須の要件であるためです。

その他の用途

詳細なセキュリティアーキテクチャガイダンスとして

モバイルアプリケーションセキュリティ検証標準のより一般的な用途のひとつはセキュリティアーキテクトのためのリソースとするものです。二つの主要なセキュリティアーキテクチャフレームワーク SABSA と TOGAF にはモバイル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完了するために必要な多くの情報がありません。MASVS を使用すると、セキュリティアーキテクトがモバイルアプリに共通する問題のコントロールを強化できるようになり、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の代替として

多くの組織は MASVS を採用することにより利益を得ることができます。2つのレベルのいずれかを選択するか、MASVS をフォークし、それぞれのアプリケーションのリスクレベルに必要なものをドメイン固有の方法に変更します。このタイプのフォークはトレーサビリティが維持されている限りにおいて奨励されています。したがって、アプリが要件 4.1 を満たしていれば、標準の派生としてフォークされたコピーにおいても同様であることを意味します。

セキュリティテスト手法の基礎として

優れたモバイルアプリのセキュリティテスト手法は MASVS に記載されているすべての要件をカバーする必要があります。OWASP モバイルセキュリティテストガイド (MSTG) では検証要件ごとにブラックボックスとホワイトボックスのテストケースについて説明しています。

自動単体テストと自動統合テストのガイドとして

MASVS はアーキテクチャ上の要件を除いて高度にテスト可能であるよう設計されています。MASVS の要件に基づいて自動化された単体、統合、受け入れテストは継続的な開発ライフサイクルに統合することができます。これにより開発者のセキュリティ意識が向上するだけでなく、結果としてアプリの全体的な品質が向上し、プレリリースのフェーズでのセキュリティテストの検証量が減少します。

セキュア開発トレーニング用に

MASVS はセキュアなモバイルアプリの特性を定義するためにも使用できます。多くの「セキュアコーディング」コースは単にコーディングのヒントを少し示すだけの倫理的ハッキングコースにすぎません。これは開発者の助けにはなりません。代わりとして、セキュア開発コースは MASVS を使用できます。トップ 10 コードセキュリティ問題などよりも、MASVS に記載されているプロアクティブなコントロールに強く焦点を当てます。

V1: アーキテクチャ、設計、脅威モデリング要件

管理目標

理想的な世界では、セキュリティは開発の全てのフェーズを通して考慮されることでしょう。しかし実際には、SDLCの後ろのステージでのみセキュリティは考慮されます。技術的なコントロールのほかに、MASVSではセキュリティが明示的に計画フェーズ中に割り当てられ、すべてのコンポーネントについて機能やセキュリティの役割が明確であることを保証する所定のプロセスを要求します。ほとんどのモバイルアプリはリモートサービスのクライアントとして動作しますので、適切なセキュリティ基準がそれらのサービスに適用されることも保証しなければいけません。モバイルアプリをテストするだけでは不十分です。

カテゴリ「V1」にはアプリのアーキテクチャと設計に関する要件が記載されています。したがって、これは OWASP モバイルテストガイドの技術的なテストケースに対応していない唯一のカテゴリです。脅威モデル、セキュア SDLC、鍵管理などのトピックをカバーするには、MASVS のユーザーはそれぞれの OWASP プロジェクトや以下にリンクされているような他の標準を参照する必要があります。

セキュリティ検証要件

MASVS-L1 および MASVS-L2 の要件を以下に記します。

#	説明	L1	L2
1.1	アプリのすべてのコンポーネントを把握し、それらが必要とされている。	✓	✓
1.2	セキュリティコントロールはクライアント側だけではなくそれぞれのリモートエンドポイントで実施されている。	✓	✓
1.3	モバイルアプリと接続されるすべてのリモートサービスの高次のアーキテクチャが定義され、そのアーキテクチャにセキュリティが対応されている。	✓	✓
1.4	モバイルアプリのコンテキストで機密とみなされるデータが明確に特定されている。	✓	✓
1.5	アプリのすべてのコンポーネントは提供する業務上の機能やセキュリティ上の機能の観点で定義されている。		✓
1.6	モバイルアプリとそれに関連するリモートサービスの脅威モデルが作られ、潜在的な脅威と対策を特定している。		✓
1.7	すべてのセキュリティコントロールは集約実装されている。		✓
1.8	暗号鍵が（もしあれば）どのように管理されるかについての明確な方針があり、暗号鍵のライフサイクルが施行されている。NIST SP 800-57などの鍵管理標準に準拠することが理想的である。		✓
1.9	モバイルアプリの更新を強制するメカニズムが存在している。		✓
1.10	セキュリティはソフトウェア開発ライフサイクルのあらゆる部分で対処されている。		✓

参考情報

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M10 - Extraneous Functionality: https://www.owasp.org/index.php/Mobile_Top_10_2016-M10-Extraneous_Functionality
- OWASP Security Architecture cheat sheet:

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

- OWASP Thread modelling: https://www.owasp.org/index.php/Application_Threat_Modeling
- OWASP Secure SDLC Cheat Sheet: https://www.owasp.org/index.php/Secure_SDLC_Cheat_Sheet
- Microsoft SDL: <https://www.microsoft.com/en-us/sdl/>
- NIST SP 800-57: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

V2: データストレージとプライバシー要件

管理目標

ユーザー資格情報や個人情報などの機密データの保護はモバイルセキュリティで重要な焦点となっています。まず、IPCのようなオペレーティングシステムのメカニズムが不適切に使用されている場合、同じデバイス上で実行されている他のアプリに機密データが意図せず晒されてしまいます。また、データはクラウドストレージ、バックアップ、キーボードキャッシュから意図せず漏れることもあります。さらに、他のタイプのデバイスに比べてモバイルデバイスは簡単に紛失したり盗まれたりします。そのため、第三者が物理的にアクセスできることがより可能性の高いシナリオとなります。その場合、機密データの取得をより困難にするために追加の保護を実装することができます。

注意: MASVS はアプリ中心であるため、MDM ソリューションにより行われるようなデバイスレベルのポリシーは対象ではありません。私たちはエンタープライズのコンテキストでこのようなポリシーを使用することにより、データセキュリティをさらに強化することを推奨します。

機密データの定義

MASVS のコンテキストにおける機密データは、以下のように、ユーザー資格情報および特定のコンテキストにおいて機密とみなされるその他のデータの両方に関係します。

- なりすまし犯罪に悪用される可能性のある個人識別情報 (PII): 社会保障番号、クレジットカード番号、銀行口座番号、医療情報。
- 被害を受けた場合に風評被害や財務コストにつながる機密性の高いデータ: 契約情報、秘密保持契約の対象となる情報、管理情報。
- 法律やコンプライアンス上の理由により保護する必要のあるすべてのデータ。

セキュリティ検証要件

大部分のデータ開示の問題は単純なルールに従うことで防ぐことが可能です。この章に記載されているほとんどのコントロールはすべての検証レベルで必須です。

#	説明	L1	L2
2.1	個人識別情報、ユーザー資格情報、暗号化鍵などの機密データを格納するために、システムの資格情報保存機能が適切に使用されている。	✓	✓
2.2	機密データはアプリコンテナまたはシステムの資格情報保存機能の外部に保存されていない。	✓	✓
2.3	機密データはアプリケーションログに書き込まれていない。	✓	✓
2.4	機密データはアーキテクチャに必要な部分でない限りサードパーティと共有されていない。	✓	✓
2.5	機密データを処理するテキスト入力では、キーボードキャッシュが無効にされている。	✓	✓
2.6	機密データはIPCメカニズムを介して公開されていない。	✓	✓
2.7	パスワードやPINなどの機密データは、ユーザーインターフェースを介して公開されていない。	✓	✓
2.8	機密データはモバイルオペレーティングシステムにより生成されるバックアップに含まれていない。		✓
2.9	バックグラウンド時にアプリはビューから機密データを削除している。		✓
2.10	アプリは必要以上に長くメモリ内に機密データを保持せず、使用後は明示的にメモリがクリアされている。		✓
2.11	アプリは最低限のデバイスアクセスセキュリティポリシーを適用しており、ユーザーにデバイスパスコードを設定することなどを必要としている。		✓
2.12	アプリは処理される個人識別情報の種類をユーザーに通知しており、同様にユーザーがアプリを使用する際に従うべきセキュリティのベストプラクティスについて通知している。		✓

参考情報

OWASP モバイルセキュリティテストガイドでは、このセクションに記載されている要件を検証するための詳細な手順を提供しています。

- Android 用 - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05d-Testing-Data-Storage.md>
- iOS 用 - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06d-Testing-Data-Storage.md>

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M2 - Insecure Data Storage: https://www.owasp.org/index.php/Mobile_Top_10_2016-M2-Insecure_Data_Storage
- CWE: <https://cwe.mitre.org/data/definitions/922.html>

V3: 暗号化要件

管理目標

モバイルデバイスに保存されるデータを保護するには暗号化が重要な要素となります。特に標準的な規則に従わない場合、物事が恐ろしく誤った方向に進んでしまうカテゴリもあります。検証されたアプリケーションが下記のような業界のベストプラクティスに従って暗号化を使用することを確認することがこの章のコントロールの目的です。

- 実績のある暗号化ライブラリの使用
- 暗号化プリミティブの適切な選択と構成
- 無作為性が必要とされる箇所での適切な乱数生成器

セキュリティ検証要件

#	説明	L1	L2
3.1	アプリは暗号化の唯一の方法としてハードコードされた鍵による対称暗号化に依存していない。	✓	✓
3.2	アプリは実績のある暗号化プリミティブの実装を使用している。	✓	✓
3.3	アプリは特定のユースケースに適した暗号化プリミティブを使用している。業界のベストプラクティスに基づくパラメータで構成されている。	✓	✓
3.4	アプリはセキュリティ上の目的で広く非推奨と考えられる暗号プロトコルやアルゴリズムを使用していない。	✓	✓
3.5	アプリは複数の目的のために同じ暗号化鍵を再利用していない。	✓	✓
3.6	すべての乱数値は十分にセキュアな乱数生成器を用いて生成されている。	✓	✓

参考情報

OWASP モバイルセキュリティテストガイドでは、このセクションに記載されている要件を検証するための詳細な手順を提供しています。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05e-Testing-Cryptography.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06e-Testing-Cryptography.md>

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M5 - Insufficient Cryptography: https://www.owasp.org/index.php/Mobile_Top_10_2016-M5-Insufficient_Cryptography
- CWE: <https://cwe.mitre.org/data/definitions/310.html>

V4: 認証とセッション管理要件

管理目標

多くの場合、リモートサービスへのユーザーログインはモバイルアプリアーキテクチャ全体の不可欠な要素です。口ジックの大半はエンドポイントで発生しますが、MASVSではユーザー アカウントとセッションの管理方法に関するいくつかの基本的な要件を定義します。

セキュリティ検証要件

#	説明	L1	L2
4.1	アプリがユーザーにリモートサービスへのアクセスを提供する場合、ユーザー名/パスワード認証など何らかの形態の認証がリモートエンドポイントで実行されている。	✓	✓
4.2	ステートフルなセッション管理を使用する場合、リモートエンドポイントはランダムに生成されたセッション識別子を使用し、ユーザーの資格情報を送信せずにクライアントリクエストを認証している。	✓	✓
4.3	ステートレスなトークンベースの認証を使用する場合、サーバーはセキュアなアルゴリズムを使用して署名されたトークンを提供している。	✓	✓
4.4	ユーザーがログアウトする際に、リモートエンドポイントは既存のセッションを終了している。	✓	✓
4.5	パスワードポリシーが存在し、リモートエンドポイントで実施されている。	✓	✓
4.6	リモートエンドポイントは過度な資格情報の送信に対する保護を実装している。	✓	✓
4.7	事前に定義された非アクティブ期間およびアクセストークンの有効期限が切れた後に、セッションはリモートエンドポイントで無効にしている。	✓	✓
4.8	生体認証が使用される場合は（単に「true」や「false」を返すAPIを使うなどの）イベントバイニングは使用しない。代わりに、キーチェーンやキーストアのアンロックに基づいている。		✓
4.9	リモートエンドポイントに二要素認証が存在し、リモートエンドポイントで二要素認証要件が一貫して適用されている。		✓
4.10	機密トランザクションはステップアップ認証を必要としている。		✓
4.11	アプリはユーザーのアカウントでのすべてのログインアクティビティをユーザーに通知している。ユーザーはアカウントへのアクセスに使用されるデバイスの一覧を表示し、特定のデバイスをブロックすることができる。		✓

参考情報

OWASP モバイルセキュリティテストガイドでは、このセクションに記載されている要件を検証するための詳細な手順を提供しています。

- 一般的に - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x04e-Testing-Authentication-and-Session-Management.md>
- Android 用 - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05f-Testing-Local-Authentication.md>
- iOS 用 - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06f-Testing-Local-Authentication.md>

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M4 - Insecure Authentication: https://www.owasp.org/index.php/Mobile_Top_10_2016-M4-Insecure_Authentication
- OWASP Mobile Top 10: M6 - Insecure Authorization: https://www.owasp.org/index.php/Mobile_Top_10_2016-M6-Insecure_Authorization
- CWE: <https://cwe.mitre.org/data/definitions/287.html>

V5: ネットワーク通信要件

管理目標

モバイルアプリとリモートサービスのエンドポイント間で交換される情報の機密性と完全性を確認することがこのセクションで説明されるコントロールの目的です。少なくとも、モバイルアプリは適切な設定でのTLSプロトコルを使用したネットワーク通信のために、セキュアで暗号化されたチャネルを設定する必要があります。レベル2にはSSLピンニングなどの追加の多層防御施策があります。

セキュリティ検証要件

#	説明	L1	L2
5.1	データはネットワーク上でTLSを使用して暗号化されている。セキュアチャネルがアプリ全体を通して一貫して使用されている。	✓	✓
5.2	TLS 設定は現在のベストプラクティスと一致している。モバイルオペレーティングシステムが推奨される標準規格をサポートしていない場合には可能な限り近い状態である。	✓	✓
5.3	セキュアチャネルが確立されたときに、アプリはリモートエンドポイントのX.509証明書を検証している。信頼されたCAにより署名された証明書のみが受け入れられている。	✓	✓
5.4	アプリは自身の証明書ストアを使用するか、エンドポイント証明書もしくは公開鍵をピニングしている。信頼されたCAにより署名された場合でも、別の証明書や鍵を提供するエンドポイントとの接続を確立していない。		✓
5.5	アプリは登録やアカウントリカバリーなどの重要な操作において（電子メールやSMSなどの）単方向のセキュアでない通信チャネルに依存していない。		✓
5.6	アプリは最新の接続ライブラリとセキュリティライブラリにのみ依存している。		✓

参考情報

OWASP モバイルセキュリティテストガイドでは、このセクションに記載されている要件を検証するための詳細な手順を提供しています。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05g-Testing-Network-Communication.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06g-Testing-Network-Communication.md>

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M3 - Insecure Communication: https://www.owasp.org/index.php/Mobile_Top_10_2016-M3-Insecure_Communication
- CWE: <https://cwe.mitre.org/data/definitions/319.html>
- CWE: <https://cwe.mitre.org/data/definitions/295.html>

V6: プラットフォーム連携要件

管理目標

アプリがセキュアな方法でプラットフォームAPIや標準コンポーネントを使用していることをこのグループのコントロールは確認します。また、コントロールはアプリ間通信(IPC)も扱います。

セキュリティ検証要件

#	説明	L1	L2
6.1	アプリは必要となる最低限のパーミッションのみを要求している。	✓	✓
6.2	外部ソースおよびユーザーからの入力はすべて検証されており、必要に応じてサニタイズされている。これにはUI、インテントやカスタムURLなどのIPCメカニズム、ネットワークソースを介して受信したデータを含んでいる。	✓	✓
6.3	アプリはメカニズムが適切に保護されていない限り、カスタムURLスキームを介して機密な機能をエクスポートしていない。	✓	✓
6.4	アプリはメカニズムが適切に保護されていない限り、IPC機構を通じて機密な機能をエクスポートしていない。	✓	✓
6.5	明示的に必要でない限りWebViewでJavaScriptが無効化されている。	✓	✓
6.6	WebViewは最低限必要なプロトコルハンドラのセットのみを許可するよう構成されている（理想的には、httpsのみがサポートされている）。file, tel, app-idなどの潜在的に危険なハンドラは無効化されている。	✓	✓
6.7	アプリのネイティブメソッドがWebViewに公開されている場合、WebViewはアプリパッケージ内に含まれるJavaScriptのみをレンダリングしている。	✓	✓
6.8	オブジェクトのデシリアライゼーションは、もしあれば、安全なシリアル化APIを使用して実装されている。	✓	✓

参考情報

OWASP モバイルセキュリティテストガイドでは、このセクションに記載されている要件を検証するための詳細な手順を提供しています。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05h-Testing-Platform-Interaction.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06h-Testing-Platform-Interaction.md>

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M1 - Improper Platform Usage: https://www.owasp.org/index.php/Mobile_Top_10_2016-M1-Improper_Platform_Usage
- CWE: <https://cwe.mitre.org/data/definitions/20.html>
- CWE: <https://cwe.mitre.org/data/definitions/749.html>

V7: コード品質とビルド設定要件

管理目標

アプリの開発で基本的なセキュリティコーディングプラクティスに従っており、コンパイラが提供する「フリー」のセキュリティ機能が有効になっていることを確認することがこのコントロールの目標です。

セキュリティ検証要件

#	説明	L1	L2
7.1	アプリは有効な証明書で署名およびプロビジョニングされている。その秘密鍵は適切に保護されている。	✓	✓
7.2	アプリはリリースモードでビルドされている。リリースビルドに適した設定である（デバッグ不可など）。	✓	✓
7.3	デバッグシンボルはネイティブバイナリから削除されている。	✓	✓
7.4	デバッグコードは削除されており、アプリは詳細なエラーやデバッグメッセージをログ出力していない。	✓	✓
7.5	モバイルアプリで使用されるライブラリ、フレームワークなどのすべてのサードパーティコンポーネントを把握し、既知の脆弱性を確認している。	✓	✓
7.6	アプリは可能性のある例外をキャッチし処理している。	✓	✓
7.7	セキュリティコントロールのエラー処理ロジックはデフォルトでアクセスを拒否している。	✓	✓
7.8	アンマネージドコードでは、メモリはセキュアに割り当て、解放、使用されている。	✓	✓
7.9	バイトコードの軽量化、スタック保護、PIEサポート、自動参照カウントなどツールチェーンにより提供されるフリーのセキュリティ機能が有効化されている。	✓	✓

参考情報

OWASP モバイルセキュリティテストガイドでは、このセクションに記載されている要件を検証するための詳細な手順を提供しています。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05i-Testing-Code-Quality-and-Build-Settings.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06i-Testing-Code-Quality-and-Build-Settings.md>

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M7 - Poor Code Quality: https://www.owasp.org/index.php/Mobile_Top_10_2016-M7-Poor_Code_Quality
- CWE: <https://cwe.mitre.org/data/definitions/119.html>
- CWE: <https://cwe.mitre.org/data/definitions/89.html>
- CWE: <https://cwe.mitre.org/data/definitions/388.html>
- CWE: <https://cwe.mitre.org/data/definitions/489.html>

V8: 耐性要件

管理目標

このセクションでは、機密データや機能を処理したりアクセスを与えたりするアプリに推奨される多層防御策を取り扱います。これらのコントロールのいずれかが欠如しても脆弱性を引き起こすことはありません。代わりに、リバースエンジニアリングや特定のクライアントサイド攻撃に対する耐性を高めることを意図しています。

このセクションのコントロールは、アプリの不正改竄やコードのリバースエンジニアリングにより引き起こされるリスクのアセスメントに基づいて、必要に応じて適用する必要があります。ビジネスリスクと依存する技術的脅威にはOWASPドキュメント "Technical Risks of Reverse Engineering and Unauthorized Code Modification", "Reverse Engineering and Code Modification Prevention"（下記の参考情報を参照）を参考にすることをお勧めします。

以下のリストのコントロールのいずれかが有効であるためには、アプリは少なくともMASVS-L1（すなわち、基本的なセキュリティコントロールが必要）のすべてとV8でのすべての低い番号の要求を満たしている必要があります。例えば、「理解の阻止」に記載されている難読化コントロールは「動的解析と改竄の阻止」、「デバイスバインディング」と組み合わせる必要があります。

ソフトウェア保護はセキュリティコントロールの代わりに使用してはならないことに注意してください。MASVS-Rにリストされているコントロールは、MASVSセキュリティ要件を満たしているアプリに、脅威に特化した追加の保護コントロールを追加することを意図しています。

以下の考慮事項が適用されます。

1. 防御されるクライアント側の脅威を明確に説明する脅威モデルを定義する必要があります。さらに、そのスキームが提供することを意図している保護グレードを明示している必要があります。例えば、明示された目標にはアプリをインストルメンテーションしようとする標的マルウェアの作者にかなりの手動リバースエンジニアリングの労力を注ぐよう強制することです。
2. 脅威モデルは合理的である必要があります。例えば、ホワイトボックス実装で暗号化鍵を隠すことは、攻撃者がホワイトボックス全体を単純にコードリフトできる場合、的外れです。
3. 保護の有効性は使用される特定のタイプの耐タンパ性および難読化のテストの経験を有する人間の専門家によって常に検証される必要があります（モバイルセキュリティテストガイドの「リバースエンジニアリング」および「ソフトウェア保護の評価」の章も参照してください）。

動的解析と改竄の阻止

#	説明	R
8.1	アプリはユーザーに警告するかアプリを終了することでルート化デバイスや脱獄済みデバイスの存在を検知し応答している。	✓
8.2	アプリはデバッグを防止し、デバッガのアタッチを検知し応答している。利用可能なすべてのデバッグプロトコルを網羅している必要がある。	✓
8.3	アプリはそれ自身のサンドボックス内の実行ファイルや重要なデータの改竄を検知し応答している。	✓
8.4	アプリはそのデバイスで広く使用されるリバースエンジニアリングツールやフレームワークの存在を検知し応答している。	✓
8.5	アプリは任意の方法を使用してエミュレータ内で動作しているかどうかを検知し応答している。	✓
8.6	アプリはそれ自身のメモリ空間内のコードとデータの改竄を検知し応答している。	✓
8.7	アプリは各防御カテゴリ(8.1から8.6)で複数のメカニズムを実装している。耐性は使用されるメカニズムのオリジナリティの量、多様性と比例することに注意する。	✓
8.8	検知メカニズムは遅延応答やステルス応答を含むさまざまな種類の応答をトリガーしている。	✓
8.9	難読化はプログラムの防御に適用されており、動的解析による逆難読化を妨げている。	✓

デバイスバインディング

#	説明	R
8.10	アプリはデバイスに固有の複数のプロパティから由来するデバイスフィンガープリントを使用して「デバイスバインディング」機能を実装している。	✓

理解の阻止

#	説明	R
8.11	アプリに属するすべての実行可能ファイルとライブラリはファイルレベルで暗号化されているか、実行可能ファイル内の重要なコードやデータセグメントが暗号化またはパック化されている。簡単な静的解析では重要なコードやデータは明らかにならない。	✓
8.12	難読化の目的が機密性の高い計算を保護することである場合、現在公開されている研究を考慮して、特定のタスクに適しており手動および自動の逆難読化メソッドに対して堅牢な難読化スキームを使用している。難読化スキームの有効性は手動テストにより検証する必要がある。可能であればハードウェアベースの隔離機能が難読化より優先されることに注意する。	✓

参考情報

OWASP モバイルセキュリティテストガイドでは、このセクションに記載されている要件を検証するための詳細な手順を提供しています。

- Android - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md>
- iOS - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06j-Testing-Resiliency-Against-Reverse-Engineering.md>

詳しくは以下の情報を参照してください。

- OWASP Mobile Top 10: M8 - Code Tampering: https://www.owasp.org/index.php/Mobile_Top_10_2016-M8-Code_Tampering
- OWASP Mobile Top 10: M9 - Reverse Engineering: https://www.owasp.org/index.php/Mobile_Top_10_2016-M9-Reverse_Engineering
- OWASP Reverse Engineering Threats -
https://www.owasp.org/index.php/Technical_Risks_of_Reverse_Engineering_and_Unauthorized_Code_Modification
- OWASP Reverse Engineering and Code Modification Prevention -
https://www.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project

付録 A: 用語集

2FA

二要素認証 (2FA) はアカウントログインに第二レベルの認証を追加します。

アドレス空間配置のランダム化 (Address Space Layout Randomization, ASLR)

メモリ破壊バグの悪用をより困難にする手法です。

アプリケーションセキュリティ (Application Security)

アプリケーションレベルのセキュリティは基礎となるオペレーティングシステムや接続されるネットワークにフォーカスするのではなく、開放型システム間相互接続参照モデル（OSIモデル）のアプリケーション層を構成するコンポーネントの分析にフォーカスします。

アプリケーションセキュリティ検証 (Application Security Verification)

OWASP ASVSに対するアプリケーションの技術的評価です。

アプリケーションセキュリティ検証報告書 (Application Security Verification Report)

対象とするアプリケーションの検証者が作成した全体的な結果と分析をまとめた報告書です。

認証 (Authentication)

アプリケーションのユーザーの申請IDを検証します。

自動検証 (Automated Verification)

脆弱性シグネチャを使用して問題を発見する自動ツール（動的分析ツール、静的解析ツール、のいずれかもしくはその両方）を使用する検証です。

ブラックボックステスト (Black Box Testing)

アプリケーションの機能を内部構造や動作に頼らずに検査するソフトウェアテストの手法です。

コンポーネント (Component)

自己完結型のコード単位です。他のコンポーネントと通信するディスクやネットワークに関連するインターフェースを持ちます。

クロスサイトスクリプティング (Cross-Site Scripting, XSS)

クライアント側のスクリプトをコンテンツに流入することを可能にする、Webアプリケーションで典型的に見られるセキュリティ脆弱性です。

暗号モジュール (Cryptographic Module)

暗号アルゴリズムを実装し、暗号鍵を生成するハードウェア、ソフトウェア、ファームウェアです。

CWE

[CWE](#) はコミュニティにより開発された一般的なソフトウェアセキュリティ脆弱性のリストです。これは共通言語、ソフトウェアセキュリティツールのものさし、および脆弱性識別、緩和、予防のためのベースラインとして機能します。

動的アプリケーションセキュリティテスト (Dynamic Application Security Testing, DAST)

実行状態のアプリケーションのセキュリティ脆弱性を示す条件を検出するように設計されたテスト技法です。

設計検証 (Design Verification)

アプリケーションのセキュリティアーキテクチャの技術的評価です。

動的検証 (Dynamic Verification)

脆弱性シグネチャを使用してアプリケーションの実行中に問題を発見する自動ツールを使用する検証です。

グローバル一意識別子 (Globally Unique Identifier, GUID)

ソフトウェアで識別子として使用される一意の参照番号です。

ハイパーテキスト転送プロトコル (Hyper Text Transfer Protocol, HTTP)

分散、協調、ハイパーメディア情報システムのためのアプリケーションプロトコルです。World Wide Web のためのデータ通信の基盤です。

ハードコードされた鍵 (Hardcoded Keys)

デバイス自体に格納されている暗号鍵です。

プロセス間通信 (Inter Process Communications, IPC)

プロセスが他のプロセスやカーネルと調和して活動するための通信です。

入力検証 (Input Validation)

信頼できないユーザー入力の正規化および妥当性確認です。

JAVA バイトコード (JAVA Bytecode)

Java 仮想マシン (JVM) の命令セットです。各バイトコードは、命令（オペコード）をあらわす1バイトもしくは場合によっては2バイトと、パラメータを渡すための0バイト以上で構成されます。

悪性コード (Malicious Code)

開発中にアプリケーションに導入された、アプリケーションの意図されたセキュリティポリシーを迂回する、アプリケーションの所有者に知られていないコードです。ウイルスやワームなどのマルウェアと同じではありません。

マルウェア (Malware)

アプリケーションユーザーや管理者の知識なしに実行時にアプリケーションに導入される実行可能コードです。

Open Web Application Security Project (OWASP)

アプリケーションソフトウェアのセキュリティを強化することにフォーカスする世界規模のフリーでオープンなコミュニティです。私たちの使命はアプリケーションのセキュリティを「可視化」することで、人々や組織がアプリケーションのセキュリティリスクについて意思決定を下せるようにすることです。参照: <http://www.owasp.org/>

個人識別情報 (Personally Identifiable Information, PII)

一人の人間を識別、接触、探索するため、あるいはコンテキストの個人を特定するために、それ自身または他の情報と共に使用することができる情報です。

位置独立実行形式 (Position-independent executable, PIE)

一次メモリのどこかに配置され、絶対アドレスに関係なく適切に実行される、マシンコードの本体です。

公開鍵基盤 (Public Key Infrastructure, PKI)

公開鍵をエンティティのそれぞれのアイデンティティにバインドする仕組みです。バインディングは認証局 (CA) における証明書の登録および発行のプロセスによって確立されます。

静的アプリケーションセキュリティテスト(Static Application Security Testing, SAST)

コーディングや設計条件のセキュリティ上の脆弱性を示すため、アプリケーションコード、バイトコード、バイナリを解析するように設計された一連のテスト技法です。SASTソリューションはアプリケーションを非稼動状態で「内側から」分析します。

ソフトウェア開発ライフサイクル (Software Development Lifecycle, SDLC)

。

セキュリティアーキテクチャ (Security Architecture)

アプリケーション設計の抽象概念です。セキュリティコントロールがどこでどのように使用されているかを特定し説明します。また、ユーザーとアプリケーションの両方のデータの場所と機密性を特定し説明します。

セキュリティ設定 (Security Configuration)

セキュリティコントロールの使用方法に影響を与えるアプリケーションの実行時設定です。

セキュリティコントロール (Security Control)

(アクセス制御チェックなどの) セキュリティチェックを実行する、もしくは(監査記録の生成などの)呼び出されたときにセキュリティ効果をもたらす、機能やコンポーネントです。

SQL インジェクション (SQLi)

悪意のあるSQL文がエントリポイントに挿入される、データ駆動型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手法です。

シングルサインオン認証 (Single Sign On Authentication, SSO Authentication)

ユーザーがひとつのクライアントにログインするとき、ユーザーが使用しているプラットフォーム、テクノロジ、ドメインに関係なく、自動的に他のクライアントにサインインするものです。例えば、google にログインすると自動的に youtube、docs、メールサービス にログインします。

脅威モデリング (Threat Modeling)

脅威エージェント、セキュリティゾーン、セキュリティコントロール、重要な技術的およびビジネス的資産を特定することにより、より洗練されたセキュリティアーキテクチャを開発することからなる技術です。

トランスポート層のセキュリティ (Transport Layer Security)

インターネット上の通信セキュリティを提供する暗号プロトコルです。

URI および URL

Uniform Resource Identifier は名前やWebリソースを識別するために使用される文字列です。Uniform Resource Locator は多くの場合リソースへの参照として使用されます。

ユーザー受け入れテスト (User acceptance testing, UAT)

本番環境のように動作するテスト環境で、実稼動前に行われるすべてのソフトウェアテストです。

検証者 (Verifier)

OWASP MASVS 要件に照らし合わせてアプリケーションをレビューしている人またはチームです。

ホワイトリスト (Whitelist)

許可されたデータまたは操作のリストです。例えば、入力の検証を実行できる文字のリストです。

X.509 証明書 (X.509 Certificate)

X.509 証明書は広く受け入れられている国際X.509公開鍵インフラストラクチャ (PKI) 標準を使用するデジタル証明書であり、公開鍵が証明書に含まれるユーザー、コンピュータ、サービスのIDに属することを検証します。

付録 B: 参考情報

以下の OWASP プロジェクトはこの標準のユーザーや採用者にとって最も役に立つものです。

- OWASP Mobile Security Project - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- OWASP Mobile Security Testing Guide -
https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide
- OWASP Mobile Top 10 Risks - https://www.owasp.org/index.php/Projects/OWASPMobile_Security_Project-Top_Ten_Mobile_Risks
- OWASP Reverse Engineering and Code Modification Prevention -
https://www.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project

同様に以下のWebサイトはこの標準のユーザーや採用者にとって最も役に立つものです。

- 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