IoTセキュリティのための設計原則トップ20

Empowering Trust®

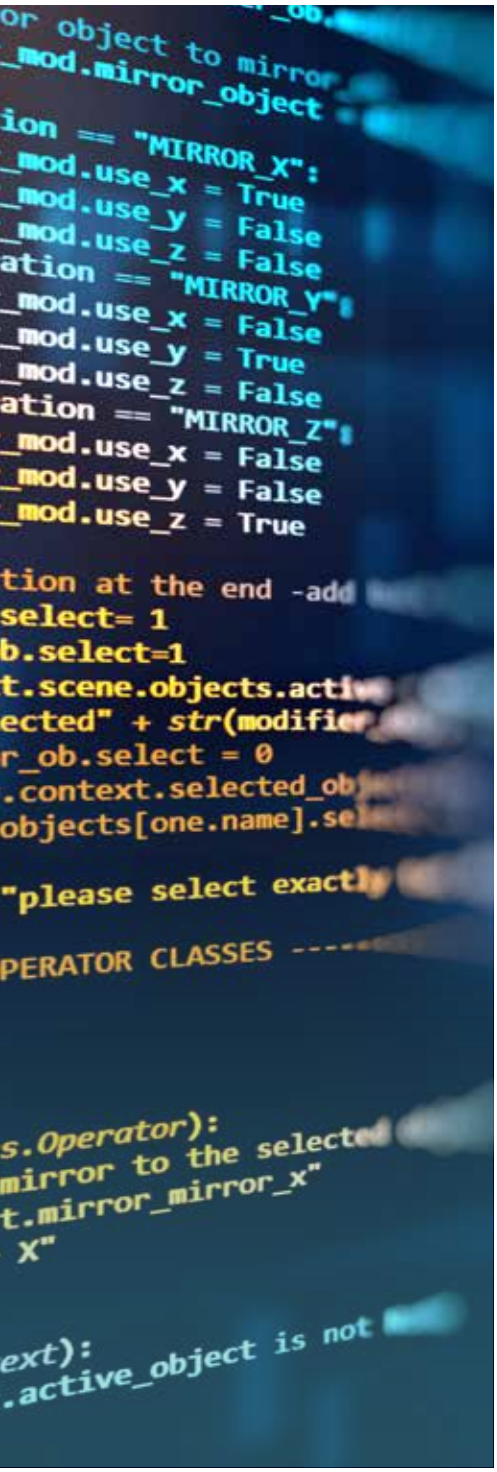
概要

競争の激しい市場では、製品を差別化する要素を見つけることが重要です。競争優位性の獲得のため、スマート機能を追加したり、製品をビジネスネットワークやインターネットに接続したりすることが多くなっています。しかし、新しい機能や接続を追加することで、システムのセキュリティが損なわれることが往々にしてあります。

さらに、コネクテッド製品のセキュリティは、組織や、さらには国家レベルでの問題となりつつあります。コネクテッドシステムの動作を制御、妨害できるマルウェアが、インターネット史上最大規模の攻撃で使用されたケースが報告されています。「コネクテッド」という性質により、クラウドサービスや消費者向けスマートフォン等それぞれのシステムで実行されるあらゆるアプリで、セキュリティを考慮することも必要になっています。

もちろん、製品開発における時間と予算の制約はますます厳しくなっており、セキュリティを組み込むことは困難が伴います。次ページでは、コネクテッドシステムのセキュリティを強化するために実施できる、シンプルなステップを紹介します。下記のステップは、最も重要な要件から順に並べられています。システム内のすべての要素、つまり製品、システム、クラウド、アプリで最優先事項として対処することが推奨されます。

優先事項 トップ5



1 安全上重要なすべての処理で手動による無効化を採用する

高度な機能は、動作している限り、有益です。しかし、システム自体に問題がなくても、このような高度な機能であっても障害が発生することがあります。顧客のローカルネットワークの構成が不適切であったり、インターネット接続が妨害されたりするケースが考えられます。このような場合には、結果として機能が失われたとしても、エンドユーザーの安全性に問題が生じないようにすることが重要です。例えば、スマートドアの施錠用に予備の物理的なキーを用意したり、IoTサーモスタットに手動による無効化や安全制限機能を備えたりすることが挙げられます。

2 システムのセキュリティを侵害する可能性のあるパラメーター（秘密/非公開暗号鍵、パスワードなど）は、デバイスごとに異なるものを設定する

セキュリティが重要な機能で使用するパスワードは、デバイスごとに異なるものを設定します。誰もが知っているパスワードは、パスワードとは言えません。しかし、これにはシンプルな解決策があります。たとえば、強力なパスワードをランダムに生成し、デバイスのシリアル番号のステッカーに印刷することができます。通常のオペレーションでデバイスに容易にアクセスできない場合は、このようなステッカーをマニュアルやクイックスタートガイドに含めておけば、ユーザーがそれを使用することもできます。もちろん、ユーザーがパスワードを忘れたり、なくしたりする可能性は常にあります。そのため、物理的なリセットボタンを押せばパスワード再設定モードに切り替わるというような、システムリカバリ方法も確実に実装することが推奨されます。

秘密（対称）または非公開（非対称）の暗号鍵も、デバイスやアプリケーションごとに異なるものを設定して管理します。セキュリティーデバイス内に暗号鍵が保管・管理されていると思われる場合でも、暗号鍵を入手する方法はいくつかあります。多くの場合、単一デバイスの鍵を抽出するために、このような方法を行う価値はありませんが、同じ鍵を数千台ものデバイスで使用している場合は、こういった攻撃の有効性は大きく異なるものになります。

3 リリース前にシステムを十分テストし、悪用の恐れがある既知の脆弱性がないことを確認する

コネクテッドデバイス、アプリケーション、クラウドサービスで使用されるソフトウェアは、多くの場合、さまざまなソフトウェアコンポーネントから構成されています。これには、既存のソフトウェア（オープンソースコードやサードパーティのライブラリなど）に加え、一般的に使用されるプロトコルや機能（データベースなど）が含まれます。このようなソフトウェアコンポーネントのそれぞれに独自の脆弱性があり、システムのリリース前に既知の脆弱性についてチェックすることが重要です。

このためには、さまざまなソフトウェアユーティリティや、クラウドベースのシステム向けのスキャニングサービスを使用します。ペイメントカード業界（PCI）では例えば、認定スキャニングベンダー（ASV）の要件に合格しているベンダーを探します。これは、スキャニングベンダーに対し、最小限の検証を行う優れた認定システムです。

4 ソフトウェアアップデートを可能にし、「インストール」および「実行」前に暗号化された認証を要求する。アンチロールバック機能を実装して、脆弱性のある古いバージョンのファームウェアをインストールできないようにする

ソフトウェアがいかに適切に設計され、テストされていようとも、バグや脆弱性が見逃されていたり、製品の出荷後に発見されることはよくあります。そのため、ソフトウェアアップデートを可能にして、このようなバグが見つかった場合に修正できるようにしておくことが重要です。しかしながら、アップデートを適切に行わないと、さらなる脆弱性が生じることになります。これを悪用して、独自のソフトウェアがデバイスにインストールされ、正常な動作を妨げる場合があります。

これを防ぐために、ソフトウェアアップデートでは暗号化された認証を行うようにします。このためには、多くの場合、システムファームウェアイメージにデジタル署名を使用し、インストール前に元のファームウェア（またはデバイスのブートローダー）で確認します。公開鍵アルゴリズム（RSAやDSAなど）に基づいたデジタル署名を使用すれば、認証データの生成に使用される鍵（非公開鍵または秘密鍵）の一部はデバイス自体で必要になりません。

代わりに、(H)MACなどの対称鍵システムを使用する場合は、この秘密鍵が各デバイスで必要となります。つまり、1台のデバイスでファームウェアにアクセスして、すべてのデバイスで使用できる有効なファームウェア署名を作成することができます（ただし、デバイスごとに異なる鍵が使用されている場合を除きますが、これはIoTシステムでは認められていません）。したがって、公開鍵暗号が強く推奨されます。

また、悪意のある者が古いバージョンのファームウェアをインストールして、新しいファームウェアでは修正されている脆弱性を復元することを防ぐ対策も講じることが重要です。このためには、ファームウェアのリリース毎に順番に番号（連番）を付け、インストールの前にこれをチェックし、インストールするファームウェアのバージョンが、現在のデバイスのバージョンより古くないことを確認できるようにします。

5 あらゆるリモート接続またはワイヤレス接続、および管理サービスの接続の認証で、ベストプラクティスのデフォルト設定を用いた業界標準のセキュリティプロトコルを使用する

TLSやWPA2などの業界標準のセキュリティプロトコルを使用して、不正アクセスや改ざんの可能性がある通信から防御することが不可欠です。さらに、セキュリティプロトコルを正しく使用することも重要です。たとえば、TLSを使用していても、適切に設定されていなければ安全でない場合もあります。このようなプロトコルでは、適切に実装されている場合にのみ、接続の認証が可能です。したがって、証明書の検証や証明書のピン留め（ピンニング）を使用して、接続が安全かつ非公開であることを確認することが推奨されます。

最新バージョンのプロトコルやソフトウェアライブラリを使用し、変更がないかをモニタリングします。これにより、問題が修正された場合にパッチを提供できます。セキュリティに関連するかどうかを問わず、可能な限りすべての通信でセキュリティプロトコルを使用します。こうすることで、攻撃者はまずセキュリティプロトコルを攻撃しなければ、デバイスに不正にアクセスすることができなくなるため、システムのセキュリティ侵害が困難になります。

この要件には、ワイヤレスプロトコルの使用も含まれます。また、WPA2などのセキュリティプロトコルも同様に重要です。顧客がセキュリティ機能を無効にできるようにすることが必要な場合もありますが、これが推奨されない理由についてガイダンスを提供することを検討してください。

残りの重要なチェックリスト

6 暗号化されていないパスワードを保存しない

パスワードの使いまわしが多く行われていることは、一般的に広く知られています。そのため、不正にアクセスした1つのシステムから入手したパスワードを、他のシステム、アカウント、オンラインサービスでも使用できる可能性があります。パスワードは、暗号化をしたうえで、BCryptなどの「一方向」かつ演算集約型のアルゴリズムで保存します。

クラウド環境では、この要素はトップ5の要件として検討する必要があります。

7 リモートアクセスとシステム管理のインターフェイスで認証を行い、セッションおよびタイムアウトの制限を設定する

ローカルネットワークの外部からデバイスにアクセスしたり、クラウドシステムにアクセスする場合など、システムへのアクセスでは、権限を与えられていない者からのアクセスを防ぐため、認証することが推奨されます。バックエンドまたはクラウドベースのシステムでは、FIDOに準拠したトークンやソフトウェアで提供されるような、2要素のセキュリティ対策を実装することを考慮します。SMSベースのワンタイムパスワードを実装することもできますが、これを悪用した攻撃が増加しているため、より新しく、安全性が確保できる手法が推奨されています。

外部システムからローカルネットワークへの接続では、VPN接続、または認証を提供する（また、必要に応じて証明書の検証/ピンニングを実行する）TLS接続をトンネルしたデータの経路制御を使用することを検討します。ローカル接続ではパスワードのみを要求し、Bluetooth/NFC接続などの物理的近接や、サービスにアクセスするための物理的なボタンによって、認証を行うこともできます。ローカルなワイヤレス接続を使用する場合は、セキュリティのベストプラクティスに従うようにしてください。

JTAGや回路内エミュレーションなどのデバッグインターフェイスは、本番用デバイスでは常に無効にします。このようなインターフェイスにアクセスするには、物理的なアクセスが必要になりますが、これらを有効にすると、悪用されて攻撃されやすくなります。

こういった管理サービスには、カメラの向きの変更から、新しい証明書、ファームウェア、その他のセキュリティ関連機能の読み込みに至るまで、幅広い機能が備えられている場合があります。リモート管理機能を介してアクセスできる複数の機能がデバイスに備えられている場合は、さまざまなレベルの認証を実装して、ユーザー機能からセキュリティ関連機能を分離することを検討してください。

さらに、1回のセッションで管理機能からアクセスできる時間に制限を設けます。こうすることで、このような機能をオンにしたまま忘れてしまうことを防ぎます。

8 暗号鍵方式で十分なランダム性を確保する

優れた乱数を生成することは、実際には非常に困難であり、安全性の低い乱数を使用することが、多くの脆弱性の原因となってきました。通常、根本的な原因として次の2つの問題があります。ひとつ目の問題はコンピューティングシステムの決定性で、これは同じプログラムで同じ入力を行うと、毎回同じ出力が生成されることを意味します。2つ目の問題は、人が乱数性が欠如していることを上手く認識できないことです。

優れた乱数を生成するために、組み込みデバイスからの入力のみに頼ることはできません。多くの場合、複数のソースからの入力を取り込むことが推奨されます。例えば、標準的な乱数機能が挙げられますが、ほかにも、A/D入力の最下位ビット、ネットワークトラフィックのタイミング、ハードディスクのシーク時間、リアルタイムのクロックソースから生成されるミリ秒データ、ユーザー入力のタイミングなども挙げられます。NIST SP 800-90Aで説明されているように、これらを組み合わせて、疑似乱数生成ツールのシードとして使用できます。

暗号鍵は、パスワードから直接生成しないようにします。強力な乱数プロセスから生成された鍵を使用するためのアクセスに、パスワードを使用する方が賢明です。

9 クラウドシステムや第三者にエクスポートされる可能性がある、すべての顧客データ（オーディオ、ビデオ、個人情報など）を詳述する。このような収集についてオプトインを提供する

多くのシステムでは、クラウド環境における処理のために高度な機能を備えています。しかし、顧客データの収集とエクスポートについて、すべての消費者が認識しているわけではありません。プライバシーの懸念から、提供される機能を希望しない顧客もいるでしょう。デフォルトでオンにするプロセスではなく、明確な情報開示とオプトインを提供することで、消費者が外部ネットワークに提供するデータをコントロールできるようにすることが重要です。

10 セキュリティプロトコルでは、業界標準の暗号化アルゴリズムと動作モードのみを使用する（ファームウェアの信頼性チェックなど）

暗号化アルゴリズムは複雑であり、今日ではどのアルゴリズムであっても安全であるとは言い切れないでしょう。アルゴリズムを安心して使用するための唯一の方法は、長年にわたる多数の専門家による調査に依存することです。それでも、新しい研究や調査で新たな欠陥が見つかることがあります。

したがって、調査で検証されており、一般的に安全性が認められている暗号化アルゴリズム、鍵長、動作モードのみを使用することが強く推奨されます。このためには、NIST SP 800 57を参照することをおすすめします。ここでは基本的に、使用するべきアルゴリズムとして、Triple DES、AES、RSA、楕円曲線暗号（ECC）のみを挙げています。

動作モードとは、データの暗号化または認証のために暗号化が実際に使用される方法であり、これも重要になります。見過ごされやすい点として、Electronic Code Book（ECB）などのシンプルな動作モードを使用すると、平文データのパターンが漏えいする可能性があり、暗号化を実装する際に意図したセキュリティが実現されないことがあります。

11 ユーザーが必要としていない、または稀にしか使用しない機能に対し、オンデマンドで有効にする機能を提供する

すべてのソフトウェアにはバグがあり、このバグの多くはセキュリティの脆弱性を招く可能性があります。製品に搭載されるソフトウェアの数が増えるほど、バグが潜んでいる可能性が高くなります。もちろん、このようなバグはテストやパッチの適用や他の方法で最小限に抑えることができますが、多くのバグはセキュリティー上の弱点を突くexploitが明らかになるまで見つからず、修正されずに残ります。多くの製品では競争の激しい市場で差別化するために、非常に多くの機能を備えるようになっています。このような相反する要件のバランスをとるため、高度な機能はデフォルトで無効にしておき、悪用が明らかになった場合でもシステムの安全を維持できるようにすることが推奨されます。

たとえば、リモートアクセス、ワイヤレスペアリング、高度な機能（メール、プリンターのインターフェイスなど）を提供している場合には、これらをデフォルトで無効にし、アクセスに時間制限を設けます。こうすることで、顧客は一定の時間のみこの機能を有効にできます。

12 ファームウェアの実行前に、コア関数と整合性を検証するパワーオン・セルフテストを実装する。可能であれば、起動中にハードウェアからの信頼できる暗号化チェーンを実装する

ファームウェアは起動するたびに検証し、インストール後から改ざんされていないことを確認することが推奨されます。これは、すべてのファームウェアで1つの署名が使用される場合に可能です。システムでシンプルな実行機能を実行している場合は、これに該当することがありますが、Linuxなど、複雑なオペレーティングシステムの場合は大幅に困難になります。さまざまなファイルやスクリプトなど、複雑なOSが正しく実行されるために必要なアイテムすべてを検証することは複雑な作業です。その解決策として考えられるのは、デバイスブートローダーを検証した後（ハードウェアのRoot of Trustから。これには、使用されるプロセッサーが対応していることが必要）、そのブートローダーを使用して、開封およびインストールされたOSイメージを検証することです。

この作業にはもちろん時間がかかります。しかし、デバイスを動作不能にし、身代金が払われるまで動作不能にするソフトウェアをインストールしようとするランサムウェアのようなものを防ぐには有効です。

13 初回操作の前に、システムのデフォルト（パスワード、証明書、鍵など）を強制的に変更させる

システムのデフォルトはできる限り使用しないことが推奨されますが、デフォルトが必要となるケースが多くあります。例えば、システムのブートストラップを初めて許可するためにはデフォルトの使用が必要になることがあります。このデフォルト値は全体的なセットアップの一環として強制的に変更させることが推奨されます。

突き詰めて言えば、デフォルトは証明書や、通常の操作のために必要なアイテムのみで使用し、インストールおよび操作の前にユーザーが変更する必要があります。これには、ファームウェアに含まれる可能性のあるテスト値も含まれます。これらは、本番システムに保管されるべきではありません。

14 エラーメッセージや、無効なメッセージへの応答で機密データを開示しないようにする

システムへのアクセスが正しく行われなかった場合、何らかのエラーメッセージが表示される事が一般的です。このようなメッセージは、システムの機密情報漏洩の原因となる可能性もあり、非常に慎重に実装する必要があります。少なくとも何らかのデバッグ状態が（当然、認証アクセスを介して）アクティブ化されるまでは、生成システムからは良好/不良のメッセージのみを示す事を検討してください。何らかのエラーが発生した場合には、復号化および検証でどのエラーが発生したかを記録するために、復号化されたデータの詳細を返したりしないでください。この場合には、単に接続を拒否するのみにとどめます。

15 暗号鍵は1つの意図した用途のみで使用する

鍵管理（暗号鍵の使用方法）は非常に重要です。暗号化アルゴリズムはセキュリティの一要素にすぎません。また、アルゴリズムの使用方法や、必要な暗号鍵の使用方法も重要です。

不正アクセスを防止するには、暗号鍵を単一の用途のみで使用することが推奨されます。データ暗号鍵は、暗号化データのみで使用します。たとえば、パスワードの保護に使用する鍵は別のものにします。ユーザーが暗号化と認証に同じ鍵（または、鍵ペア）を使用しないようにしてください。1つの鍵は単一の用途に使用します。

16 すべてのシステムで最小限の権限を実装する

最新のプロセッサーでは多くの場合、複数の権限レベルが提供されており、この中にはメモリや、その他リソースへのアクセスが可能なものもあります。これらの機能はソフトウェアにより使用されて、適切な権限を持つソフトウェアのみがアクセスできるようにすることで、デバイス内の資産を保護することができます。プロセッサーのハードウェアレベルの権限制御インターフェイスは、多くの場合、LinuxなどのOSによって管理されています。しかし、これはデバイスに複雑なOSが搭載されていない場合（たとえば、シンプルな実行機能を使用する場合や、代わりに縮小版RToSを使用する場合など）でも制御できます。

可能な限り、組み込みシステムでのルート権限や監督者レベルの権限の使用は最小限に抑え、暗号鍵などの機密データを最高レベルの権限で維持してください（最もアクセスしにくくする）。アプリやクラウドベースのシステムでも同様です。高レベルの権限の使用は最小限に抑え、それぞれのユーザーまたは権限セットとして、分離するようにします。

17 データメモリの実行を防ぐ保護を実装する

リモート攻撃の多くでは、特定の脆弱性を利用して悪意あるコードを実行しようとします。コード内のすべての脆弱性を防ぐことはほぼ不可能ですが、さまざまな方法で、コーディングの脆弱性を利用してリモートでコードが実行される可能性を回避することは可能です。

多くのプロセッサーでは、「実行不可」機能が備えられています。これは、直接実行可能なコードを参照するためには使用できない、メモリの特定領域を指定するものです。ほかにも、コード用とデータ用のメモリ領域を完全に分離しているプロセッサーもあります。これにより、直接的なコードインジェクション攻撃が不可能になります（ただし、他の種類の攻撃を受ける可能性はあります）。

直接的なハードウェア保護に加えて、ソフトウェアレベルで適用できる保護もあります。このような保護は、OS自体から提供されることも、デバイスに読み込むオブジェクトコードを作成する際にコンパイラー設定を介して適用されることもあります。

システムのさまざまなコンポーネントで、どのような保護が可能であるかを判断し、技術的および運用上で可能な限りの対策を実装してください。

18 デバイスについて定義された機能の範囲にない、外部から提供されるコマンド、スクリプト、その他のパラメーターの直接的な実行を許可しない

直接的なコード実行に加えて、他の解釈プログラムや実行環境が利用できる場合には、これを悪用して他の方法でもシステムに不正にアクセスすることができます。たとえば、システムでJavaScriptなどの非ネイティブコードの実行を許可していることがありますが、これは脆弱性につながる可能性があります。常に、JavaScriptのような機能の無効化が必要であるというわけではありませんが、この種の機能を使用する必要性について理解しておくことが推奨されます。このような機能を含めるには、多くの場合、システムの全体的なセキュリティ体制を維持するため、より複雑なセキュリティソリューションが必要になります。

19 定義済みの機能に必要なコードとシステムのみを含めるようにデバイスのファームウェアを作成し、コンパイルする。本番用コードを作成する際には、常にデバイス内のデバッグおよび開発用機能を削除するか、無効にする

コードが長くなればなるほど、セキュリティ上の欠陥が潜んでいる可能性は高くなります。したがって、製品出荷後に欠陥が発覚し、パッチやその他の対応策が必要になる確率を低くするために、できる限りコードを削除しておくことが賢明です。これには、通常は実行されないコードも含まれます。コードが使用されない場合でも、デバイス内に含めることで、将来的にセキュリティの問題を引き起こす可能性があるからです。

同様の理由で、出荷前にはデバイスからデバッグおよび開発用コードを削除しておくことが重要です。このためには、多くの場合「ifdef」ステートメントを隔離します。こうすることで、コンパイル中にこのような機能が自動的に削除されます。製品に問題が発生した場合に備えて、コードにこれらの機能を含めておきたいと考えることは理解できますが、このような機能によって、エンドユーザー環境で通常では提供されないアクセスや情報にアクセスできるようになり、脆弱性の元となる可能性があります。

20 脆弱性管理プログラムを実装し、リリース前から製品ライフサイクルの終わりまで、製品のセキュリティ上の欠陥を定期的にモニタリングし、対処する。顧客にパッチを配布し、情報を提供するプロセスを含める

セキュリティとはムービングターゲットであり、100%安全なシステムはありません。新しい脆弱性や攻撃手法が発見されるたびに、新旧両方のシステムがこれらのセキュリティ上の欠陥に影響を受けないようにするプログラムを用意することが重要です。これを可能にするには、システムセキュリティの継続的なモニタリングと更新を確実にするための、経営幹部が承認/実施するプログラムが必要です。

パッチが作成されず、顧客のシステムにインストールできなければ、システムにパッチをインストールできる機能があったとしても、価値が無いものになってしまいます。

ULのサイバーセキュリティサービスの詳細については、
UL.com/cybersecurityをご参照下さい。
お問い合せ先：ULCyber@ul.com

UL.com/cybersecurity

CT0119