エッジのインテリジェンス化 Part4：エッジ・ノードのセキュリティ

著者：Ian Beavers、Erik MacLean
Analog Devices, Inc.

IoT（Internet of Things）に対応するシステムに向けた攻撃について、大々的に報じられるケースが増えています。セキュリティの観点から言うと、脆弱性を抱えたネットワーク、エッジ・ノード、ゲートウェイは決して少なくないことが、絶えず白日の下にさらされている状態にあります。最近では、Miraiを使って構築したボットネットが話題になりました。telnetサーバのパスワードが、デフォルトの文字列のまま変更されていないケースがあります。Miraiを利用したボットネットは、そのようなサーバが稼働するデバイスにログインし、250万を超えるIoT対応ノードを感染させました[1]。その後、DoS（Denial of Service）攻撃がサーバに仕掛けられ、インターネットに対する極めて多くのアクセスに対して被害が及びました。また、Reaperを利用したボットネットが、ソフトウェアの脆弱性を突いて感染し、100万台を超えるIoTデバイスに攻撃を加えるという事態も生じました。インターネットに接続された水槽がカジノのネットワークへの侵入口となり、10GBものデータが盗まれたこともありました。更には、スマート・テレビも標的となっており、諜報や監視といった目的で悪用されています。

組み込みセンサーを使用するシステムがインターネットに接続され、様々な危険にさらされるようになったのは、つい最近のことです。IIoT（Industrial IoT）の構成要素であるそれらのセンサーには、ウェブ・サーバのように、悪意と向き合いながら進化してきた20年もの歴史といったものは存在しません。そのため、IIoTシステムに対する攻撃の多くは、1990年代かそれ以前に一般的に見られたものであることが確認されています。一般に、IIoTシステムの寿命は、従来のコンピューティング・システムと比べてはるかに長くなります。デバイスによっては、配備されてから数十年にもわたって稼働し続ける可能性があります。しかも、どのようなスケジュールで保守が行われるかわからないケースも存在します。

PCやサーバは、セキュリティ対策に費やすことが可能な能力を備えています。それに対し、IIoTノードは、一般的に処理能力も消費電力も低く抑えられています。そのため、セキュリティ対策に割り当てられる消費電力もわずかになります。開発コストにも限度があるため、往々にしてセキュリティ対策は他の要素との兼ね合いで行われます。IIoTシステムについては、民生向けのIoTシステムよりも開発にコストをかけられる可能性があります。それでも、スケーラビリティを持たせるためのコストがかかるという問題を抱えています。セキュリティを無視すると、その隠れた影響が製品が配備された後に現れることがあります。その影響に対処するために、結局は多大なコストをかけなければならなくなるといった事態に陥りかねません。

IIoTシステムに使用されるデバイスは、センサーやアクチュエータにより、物理的な世界と相互にやり取りします。これまでのところ、サイバー攻撃の被害は主にデータの喪失のみにとどまっています。しかし、IIoTシステムをハッキングした場合、従来よりも容易に物理的な世界に侵入することができます。つまり、サイバー攻撃によって、物理的な被害が引き起こされるおそれがあるということです。そのような物理的な被害は深刻なものであり、1つの問題によって数百万米ドル規模の産業プロセスの停止や破壊が生じる可能性があります。場合によっては、人命にかかわる問題につながってしまうかもしれません。

ネットワークに接続された世界

通常、IIoTデバイスは何らかのネットワークに接続されます。多くの場合、インターネットに接続されることになります。この接続性が、それらのデバイスを危険にさらす最大の要因です。伝染病と同じように、他のマシンとの接触によって感染は拡大します。攻撃ベクトルは、システムが外部の世界とやり取りする個所に存在します。攻撃者がシステムに作用を及ぼすことができるのは、そのシステムがネットワークに接続されているからに他なりません。システムを設計する際には、セキュリティに関連するそもそもの疑問として「このデバイスは本当にネットワークに接続する必要があるのか」と問う必要があります。セキュリティに関するリスクは、デバイスをネットワークに接続することによって劇的に高まるからです。

システムを保護するための最良の方法は、ネットワークに接続しないことです。あるいは、接続先をクローズド・ネットワークに限定すればよいということになります。多くのIIoTデバイスは、「接続できるから」というだけの理由でネットワークに接続されています。そのデバイスをネットワークに接続することによって、セキュリティ面でのリスクに勝るメリットが得られるのかと問う必要があります。危険が及ぶ範囲は、インターネットに接続されたシステムだけにとどまりません。そのシステムとやり取りする他の従来型システムも危険にさらす可能性があります。

それ自体は安全なネットワークやノードが存在したとします。多くの場合、それらと、セキュリティの面では、はるかに劣る従来型のネットワークとの間で相互運用を図る必要があります。つまり、最も大きなセキュリティ上のリスクは、IIoTシステムの外側に存在するかもしれないという状態になります。その場合、ネットワークの内部でIIoTシステムを保護する必要があります。

ノードにおけるセキュリティに関して考慮すべき事柄[2]

- 機密保持：なりすましによる攻撃などを防ぐために、権限のないユーザにはデータが開示されないように保護します。
- 認証：デジタル証明書を使用し、2 つのマシンの間でアイデンティティを確認します。
- セキュア・ブート：ROM に格納されたブートローダによって、第 2 ステージ・ブートローダが真正であることを確認します。
- ファームウェアのセキュアなアップデート：メーカーが認証済みのコードだけを許可します
- 承認：真正なノードだけがネットワークにアクセスできるようにします。
- 完全性：データを改ざんから保護します。
- アカウンティング：データ、ノード数、タイムスタンプを適切にアカウンティングすることによって、IIoT 対応のネットワークに対する望ましくないアクセスを防ぎます。
- セキュアな通信：低消費電力のノードに適用できる暗号化プロトコルを使用します。
- 可用性：ユーザの希望に応じて確実にアクセスできるようにします。
- 否認防止：真正な通信要求が拒否されないように保証します。
- 信頼性：電気的に過酷な環境でも、信頼性の高いアクセスを維持します。

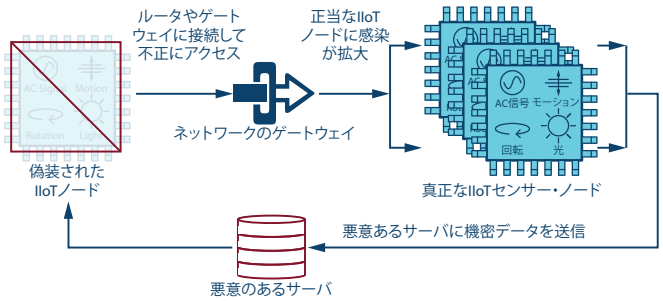

図1. 既知のノードへのなりすまし。
不正なゲートウェイへのアクセスが行われています

隔離

システムを互いに隔離すれば、アタック・サーフェス（攻撃の対象となる領域）を減らし、マルウェアの拡散を制限することができます。具体的には、ネットワークに接続されているシステムから、ネットワークへの接続が不要なシステムを隔離します。また、隔離用のエアギャップを設けたり、リスクの大きいシステム用のネットワークから隔離された別のネットワークを厳しく監視したりすることを検討します。重要なシステムについては、外部から完全に隔離することができれば理想的です[3]。

コネクテッド・カーのインフォテインメント・システムは、これまで見たことのないような数多くの攻撃ベクトルに車両をさらしてしまう原因になる可能性があります。メインのエンジン制御ユニットは、インフォテインメント・システムとは何の関係もありません。したがって、同システムを介して同ユニットにアクセスする手段を設けてはなりません。通常、車両には2つの独立したCAN（Controller Area Network）バスが存在します。それらによって、最も重要なシステムがそれ以外の部分から分離されます。ただ、それでも両者は何らかの方法で互いに接続されています。結果として、一方に不正にアクセスすることで、他方を不正に制御するといったことが行われてしまうかもしれません。それらのネットワークの間を完全に隔離すれば、命にかかわるようなものから、それよりもはるかに軽微なものまで、あらゆる不正アクセスのリスクを軽減することができます。

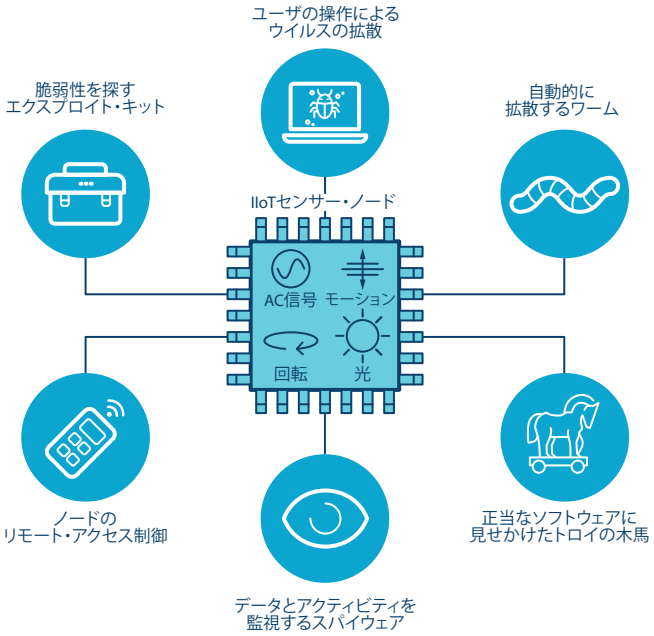

図2. IIoTシステムに感染する可能性のある多様なマルウェア

エッジへの処理の移行

IIoTシステムの多くはクラウド・サーバに接続されます。クラウド・サーバは、デバイスから送信された情報を収集して処理すると共に、デバイスの管理も担います。しかし、デバイスの数が増加していくと、クラウドでそれらすべてを管理するのは難しくなる可能性があります。そこで、多くのシステムでは、クラウドに対するトラフィックを減らすために、エッジのIIoTデバイスに処理を移行するということが行われています。

多くの場合、データは一種のアセットとして捉えられています。大規模なデータ・セットに隠されたパターンを明らかにするために、データのマイニングや販売が行われます。ただし、一般的に、収集されたデータの大部分はほとんど利用価値がありません。ところが、攻撃者にとってそれらは利用できるものである可能性があります。機密データは攻撃者にとって標的になり、不利益を引き起こす要因になります。収集したデータは、必要なものだけに絞り込み、不要なデータはできるだけ早く破棄する必要があります。それはセキュリティの強化につながるだけでなく、収集したデータの有用性の向上にもつながります。機密データとなる可能性が高い情報を特定し、それらに対するアクセスを制限することが重要です。

データをエッジで処理すれば、クラウドに送信／開示されるデータ量を減らすことができます。データを多くの場所に送信するほど、その機密性を保つのは困難になります。データをノードに送信するたびに、そのノードが新たなアタック・サーフェスになります。すなわち、不正にアクセスされてデータが漏洩するおそれが高まるということです。また、アタック・サーフェスは、指数関数的に拡大する可能性があります。

機密データをエッジに配置すれば、その機密データに対するアタック・サーフェスを制限することができます。1つのエッジ・ノードにしかデータが存在しなければ、盗まれる確率は大幅に低下します。例えば、駐車スペース用の占有センサーを使う場合、動画をストリーミングすることなく、画像処理後のバイナリ信号によって、車両の有無の情報だけを報告するようにします。画像に含まれる不要な大量のデータは破棄します。それによって、受信側のサーバの負荷は軽減され、データが悪意のある監視目的に再利用されるおそれはなくなります。

民生用のIoTシステムと同様に、IIoTシステムにも、以下のような保持する必要のあるプロプライエタリな情報や機密情報が存在します。

- プロプライエタリなアルゴリズム
- 組み込みファームウェア
- 顧客の情報
- 財務の情報
- アセットの場所の情報
- 装置の使用パターン
- 競合するインテリジェンス
- より大規模なネットワークへのアクセス

フォグを介した接続

IIoTデバイスの中には、エッジベースで処理を行うだけの能力を備えていないものがあります。別の新しいトポロジとして、クラウドベースのシステムとエッジベースのシステムのハイブリッド形式であるフォグ（霧）モデルも存在します。フォグ・モデルにおいて、エッジ・ノードはまずゲートウェイに接続されます。ゲートウェイはデータを受信し、何らかの処理を施してからクラウドに送信します。1台のゲートウェイによって、多数のIIoTデバイスに対応することが可能です。ゲートウェイは、バッテリで動作するものである必要はありません。そのため、電力の面で制約のあるIIoTデバイスよりもはるかに高い処理能力を持たせることができます。ただ、IIoTデバイスよりもコストは高くなります。

フォグ・モデルは、拡張性の問題から考案されたものですが、セキュリティの面でも有用である可能性があります。ゲートウェイのデバイスは、制約が大きいために独自にセキュリティを確保するための機構を設けることのできない脆弱なエッジ・ノードの保護に利用できる可能性があります。ただ、独自のセキュリティ機構を設けられないとはいえ、何もしないよりもある程度の対策を施しておく方がよいでしょう。そこで、個々のノードを直接管理するのではなく、ゲートウェイを利用してその配下にあるすべてのノードを管理するという方法が考えられました。フォグ・モデルでは、サービスの中断を回避しつつ、IIoTシステムにおける問題に対応することも可能です。例えば、ミッション・クリティカルな製造ラインを停止させることなく、ゲートウェイとのやり取りによって、セキュリティの問題に対処できる可能性があります。

プロビジョニングと配備

IIoTの大きな課題の1つは、いかにして多数のデバイスを配備し、管理するのかということです。一般に、IIoTシステムは広い範囲にわたって構築されます。そのため、設定と構成（コンフィギュレーション）が非常に困難です。また、システムの寿命が長いので、あるチームが配備したIIoTシステムが何年も稼働し続け、別のチームがサポートを担当しなければならなくなるケースもあります。

IIoTシステムは、デフォルトでは脆弱な認証メカニズムしか備えておらず、セキュリティが十分に確保されていないことがよくあります。Miraiによるボットネットの例からもわかるように、ほとんどのユーザは、IIoTデバイスにログインして設定を行うことはありません。設定できることさえ知らない可能性があります。IIoTシステムのほとんどのユーザは、デバイスは箱から取り出すだけでそのまま使用できると考えています。これに対応するためには、システムはデフォルトでセキュリティが確保された状態になっていなければなりません。ユーザは、デフォルト以外の状態にデバイスを設定することはないという想定が必要です。その意味で、デフォルトのパスワードとして脆弱な文字列を設定するのは、よくある過ちだと言えます。

ネットワークのセキュリティ

IIoTシステムにおいて最も注目されるのはエッジです。しかし、実際にはクラウドやサーバを軽視しないことが重要になります。クロスサイト・スクリプティング、SQLインジェクション、クロスサイト・リクエスト・フォージェリなど、サーバ側の一般的な脆弱性についてテストを実施し、API（Application Programming Interface）に脆弱性がないかどうかを確認します。その上で、サーバ上で実行されているソフトウェアには、パッチが必ず速やかに適用されるようにします。

ネットワーク上では、伝送中のデータを保護する必要があります。そうしなければ、データが傍受されたり悪意を持って改ざんされたりするおそれがあるからです。そこで、TLS（Transport Layer Security）やSSH（Secure Shell）などのセキュアな暗号化プロトコルを使用して、伝送中のデータを保護することになります。理想的には、データはエンドtoエンドで保護するべきです。

多くの場合、IIoTに対応するネットワークでは、周辺との境界が曖昧になります。通常、IIoTのセンサー・ノードは、ネットワークの末端に配置されます。しかし、それらは固定のゲートウェイを通して、より大規模な産業用ネットワークに簡単にアクセスできるポータルでもあります[4]。ネットワークにおいては、そのようなデバイスの認証を適切に行う必要があります。そのことが、悪意ある第三者によってトラフィックが改ざんされるのを防ぐことにつながります。

ネットワークにおいてデータのトラフィックを保護するには、セキュアな通信プロトコルを使用しなければなりません。ベスト・プラクティスは、セキュアであることが既知の標準プロトコルを使用することです。イーサネットを使用したLAN上のセキュリティは、IEEE 802.1AE MACsecを使用することで確保できます。ワイヤレスLANには誰でも簡単にアクセスできるので、よりリスクが大きいと言うことができるでしょう。WPA2（Wi-Fi Protected Access II）は、IEEE 802.11をベースとするワイヤレス・ネットワークにセキュリティを提供します。ワイヤレスのIIoTソリューションでは、消費電力を抑えられるIEEE 802.15.4がよく利用されます。同規格は、セキュリティの確保に向けた一連のプロトコルを独自に定義しています。それらはレイヤ2のプロトコルであり、LAN上のトラフィックのみを保護の対象とします。

インターネットなど、LANの外側にも伝送されるトラフィックを保護するには、エンドtoエンドでセキュリティを確保できるより高いレイヤのプロトコルが必要です。インターネットでは、トラフィックの保護にはTLSが一般的に使用されています。これであれば、エンドtoエンドでセキュリティを確保できます。TLSはTCP（Transmission Control Protocol）を使用します。一方、多くのIoTデバイスは、UDP（User Datagram Protocol）を使用します。UDPに対応するプロトコルとしてはDTLS（Datagram Transport Layer Security）が挙げられます。IoTデバイスには、消費電力とメモリの面で制約があります。TLSであれば、そうした制約の厳しいアプリケーションに対し、最小限の努力で適用することができます。より制約の厳しいデバイス向けには、IETF（Internet Engineering Task Force）が新たにCoAP（Constrained Application Protocol）という規格を策定中です。

エンドポイントのセキュリティ

伝送中のデータを保護するのは重要かつ必須です。ただ、現実の攻撃は、一般的にはエンドポイントを標的として行われます。ネットワークに接続されるインターフェースについては、脆弱性に対する保護を強化する必要があります。IIoTシステムにおいてセキュリティを確保するための1つの方法は、センサー・ノードのデバイスに保護機構を組み込むことです。それが最初の重要なセキュリティ・レイヤとなり、企業のファイアウォールがデバイスを保護するための唯一の砦という状態を回避できます。遠隔地に配備される企業用のモバイル・デバイスやIIoTセンサーにおいては、特にそのことが重要になる可能性があります。

IIoTデバイスのセキュリティに向けたソリューションは、多様なサイバー攻撃からの保護を実現できるものでなければなりません。デバイスのファームウェアが改ざんされていないことを確認すること、デバイス内に格納されたデータを保護できること、送信／受信に対する保護が行えること、サイバー攻撃を検出して報告できることが求められます[5]。設計初期の段階からセキュリティについて考慮していなければ、これらを実現することはできません。

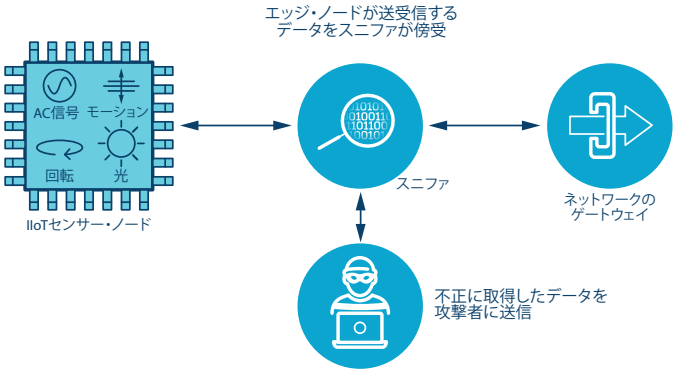

図3. 中間者攻撃の概念。悪意あるアクセス・ポイントがノードとゲートウェイの間に挿入されます。

組み込みデバイスに対する万能なセキュリティ・ソリューションというものは存在しません。もちろん、セキュリティに関する一般的なフレームワークを提供するソリューションは存在します。しかし、完全なフレームワークを構築するには、特定のデバイス、ネットワーク、システム全体の保護に必要な中核的な機能について検討しなければなりません。また、重要なセキュリティ機能を確保しつつ、任意の具体的な要件に応じてカスタマイズを施せる柔軟性の高いソリューションが必要です。

IIoTシステム向けの“オートクレーブ”

病気の蔓延を防止しつつ、手術器具を再利用するためには、殺菌処理が不可欠です。代表的な殺菌処理装置としてはオートクレーブが挙げられます。これを使用すれば、高温、高圧の蒸気によって、素早く器具の殺菌を実施できます。つまり、すべての菌を死滅させ、器具を良好な状態に戻すことが可能だということです。例えば、手術に使用したメスも、殺菌処理を施すことにより、安全に再利用できるようになります。

それと同様に、不正なアクセスが行われた後に、システムを既知の良好な状態に戻す能力も重要です。これは、あらゆる攻撃に対して防御を固めることよりも重視すべきことかもしれません。レジリエントなシステムであれば、素早く確実に復旧して動作を再開することができます。

では、どのようすれば、一度感染したシステムを感染前の状態に戻すことができるのでしょうか。何らかの未知の方法で状態が改変されたシステムを、どのようにすれば元に戻せるのかということです。例えば、リモート・エクスプロイトでは、プロセッサの制御能力が奪われ、悪意のあるコードがシステムに挿入されます。何らかの方法によってファームウェアを改変するか、マルウェアへのすり替えを行うことにより、システムの動作が変更されるということです。一度これが行われると、プロセッサはもはや信頼できる状態ではなくなります。

一般に、組み込みシステムを、感染した状態から確実に回復させるのは容易なことではありません。システムを感染前の状態に戻し、実際にクリーンであることを確認するための唯一の方法は、不揮発性メモリに格納されたデータを外部のリーダーに直接ダンプすることです。そうすれば、元のファームウェアとの比較により検証を実施することができます。その結果、改変が認められれば、本来のファームウェアのデータで置き換えるということで対処できます。しかし、ほとんどのシステムは、このようなことが行えるようには設計されていません。

システムの完全性を保つための1つの方法は、機械式スイッチによって物理的に不揮発メモリに対する書き込みを行えなくすることです。スイッチを切り替えて、書き込みを禁止する状態に設定することにより、ハードウェアによってメモリを物理的に保護するということです。プロセッサのドメインの外部にメモリの制御を移せば、デバイスに物理的にアクセスすることなく、永続的なマルウェアをメモリに対して遠隔からインストールすることは不可能になります。その結果、攻撃を仕掛けてくる可能性のある人は、インターネットによってつながっているすべての人から、デバイスに長時間にわたって物理的にアクセスできる人に限定されます。ファームウェアのアップデートは、極めてまれにしか実行されません。ファームウェアのアップデートが必要な場合には、メモリに対する書き込みが可能な状態にスイッチを設定してアップデートを許可します。アップデートが完了したら、再び書き込みが行えない状態に戻します。

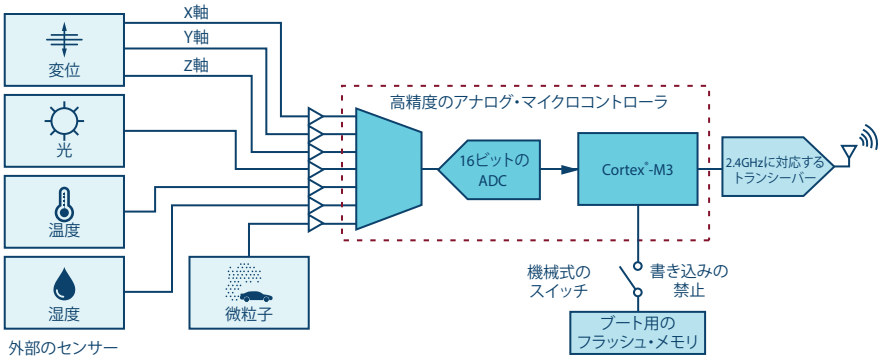

図4. 機械式スイッチによる保護。アップデートを実行するとき以外、ファームウェアの書き換えは物理的に行えない状態にします。デバイスの完全性を保つための有効な手段です。

多くのデバイスにおいて、不揮発性メモリは、書き込みを行うためのアクセスに必要なデータを格納するためにも使用されます。よりセキュアなシステムでは、ソフトウェアではなくデータを格納するために、独立した不揮発性メモリが使用される場合があります。そのメモリに悪意のあるデータが書き込まれると、ソフトウェアのバグが利用されるという形で、システムが攻撃されてしまうかもしれません。したがって、システムを徹底的に解析／テストし、このメモリにどのようなデータが格納されていたとしても、システムに被害が及ばないようにする必要があります。メモリ・チップを追加するとコストが増加しますが、フラッシュ・メモリの中には、一部のセクタだけ書き込みを禁止し、その他の部分は書き込みを許可するということが可能なものもあります。

セキュア・ブート

セキュア・ブートは、ブートの実行時に不正なソフトウェアがデバイスに読み込まれることを防ぐためのものです。信頼性を保つためのチェーンの先頭に位置するものだと言うこともできます。セキュア・ブートは、ノード上にある不揮発性メモリの読み取り専用領域に記録された第1ステージ・ブートローダによって開始されます。このブートローダの役割は、第2ステージ・ブートローダが真正なものであることを確認することだけです。その後の処理は、第2ステージ・ブートローダが引き継ぎます[6]。第2ステージ・ブートローダは、一般的に第1ステージ・ブートローダよりも複雑ですが、再プログラムが可能なフラッシュ・メモリに格納できます。第2ステージ・ブートローダは、OSやロードされたアプリケーションが、信頼できるソースから得た有効なものであるか否かを確認します。

セキュア・ブートとファームウェアのセキュアなアップデート機能を備えるIIoTノードは、改ざんされたコードや悪意のあるコードではなく、正規のコードがデバイスで実行されることを保証します。また、マルウェアや悪意のあるコードが永続的にインストールされることを防ぎます。デバイスは、改変されていない正規のコードのみを実行するか、ブートに失敗するかのいずれかとなります。

通常、セキュア・ブートのプロセスは、コードの真正性を保証するためにデジタル署名を拠り所にします。デバイスの製造時には、組み立ての工程において、OEMの秘密鍵を利用した署名がコード・イメージに組み込まれます。ノードは、OEMの公開鍵を使用することにより、ファームウェアのイメージの署名を確認します。

コードの保護は、対称鍵暗号を用いたメッセージ認証コード（MAC：Message Authentication Code）によって行うことができます。そのためには、秘密鍵をデバイス上に格納する必要があります。結果として、そのデバイスはリスクを抱えることになります。ただ、MACを使用すると演算は容易になります。

セキュア・ブートを利用すれば、安全性の向上を図ることができます。ただし、そのことがエンド・ユーザに制約を課すことになる場合もあります。エンド・ユーザは、デバイス上で実行されるソフトウェアを変更したり、独自のソフトウェアを実行したりすることができないからです。アプリケーションによっては、ユーザがブートについてある程度設定できるような柔軟性を確保する必要があるかもしれません。つまり、ユーザ独自のコードは信頼に足るものだといった設定が行えるようにするということです。

セキュア・ブートと同様に、ファームウェアのセキュアなアップデートでは、その実行時に新しいコード・イメージに含まれるOEMの署名を確認します。ダウンロードされたイメージが有効なものでない場合には、それらを破棄してアップグレードを中止します。有効なイメージだけを受容して、デバイスのメモリに保存します。

脆弱性は、必ずどこかに潜んでおり、いずれは発見されるものだと想定しなければなりません。言い換えれば、脆弱性が発見されたり悪用されたりした場合の対処法を事前に計画しておく必要があります。一般的には、脆弱性を修正するためのソフトウェアのアップデート版やパッチをデバイスにインストールするための手段が必要です。また、アップデートのプロセスも、誰でもデバイスにマルウェアをインストールできるような攻撃ベクトルとして利用されないように、適切に実装する必要があります。パッチをインストールできるようにするためだけに、ネットワーク経由でデバイスにアクセスできるようにしていたのでは、リスクを緩和する効果よりも、リスクを招くおそれの方が大きくなってしまいます。

セキュアな通信

ほとんどの技術者は、SSL（Secure Socket Layer）/TLS、SSH、IPsecといった通信プロトコルによって、セキュリティを確保しようと考えます。その結果、多くの組み込みデバイスには、セキュアな通信機能が追加されています。しかし、新たな侵入経路となる別の攻撃ベクトルが存在し、それがセキュリティ上の脅威になります。多くのIIoTセンサー・ノードは、低消費電力のプロセッサによって消費電力を抑える構成で動作します。ただ、低消費電力のプロセッサは、TLSやIPsecといった優れた選択肢のすべてをサポートするとは限りません。セキュリティに対応するプロトコルは、セキュアなデバイスを構築するための適切な出発点です[7]。そうしたプロトコルは、パケット・スニッフィング、中間者攻撃、反射攻撃（リプレイ・アタッ

ク）、そしてノードと不正に通信しようとする試みから保護できるように設計されています。

多くの場合、IIoTシステムのエッジに配備される小型のセンサー・デバイスには、ZigBee、Bluetooth® Low Energy（BLE）といったワイヤレス・メッシュ・ネットワークのプロトコルが適用されます。そうしたプロトコルには、セキュリティを確保するための仕組みが盛り込まれています。ただし、その能力は万全ではなく、比較的脆弱なものだと言えます。既に多くのエクスプロイトが公開されており、高度なハッカー（クラッカ）はそれらについて熟知しています。通常、小型化が図られたIIoTデバイスは、TLSやIPsecをサポートしない非常に低コストで低消費電力のプロセッサによって動作します。そうした小型のエッジ・デバイスの場合、UDP上で動作するTLSであるDTLS（Datagram Transport Layer Security）を利用することによって、セキュアな通信を実現することができます。

物理的なセキュリティ

IIoTシステムのエッジに配備されたハードウェア・ノードやゲートウェイ、あるいはフロントエンドのセンサーを標的として、物理的な攻撃が仕掛けられることもあります。通常、そうした攻撃を行うには、システムに物理的にアクセスする必要があります。その被害は、IIoT対応のハードウェアの有効性を抑えるレベルにとどまるケースもあります。攻撃者はノードに対する改ざん行為を行い、IIoT環境内にあるセンサーなどのデバイスを制御できない状態にします。続いて、機密データや組み込み済みのファームウェアのコードをソースから抽出します。悪意のあるノードを挿入し、IIoTネットワークの正規のノードの間に配備するといったことが行われます[8]。

このような攻撃を緩和するために、設計の段階でハードウェア上に事前の対策を複数盛り込むということが行われます。例えば、リード付きのデバイス、露出した銅製のビア、未使用のコネクタなどは、最小限に抑えるか、回路から完全に排除します。そうすることで、それらを利用した物理的な方法で信号をプローブするということが行えないようにします。部品の詳細を示すことになり、ハッカーに対して追加の情報を与えるおそれのあるシルク・スクリーンは、絶対に必要というわけでなければ取り除きます。産業用のコンフォーマル・コーティングは、システムの複雑さを増大させるかもしれません。それでも、ハードウェアと各要素の間の緩衝材として用いるだけでなく、プリント回路基板上の各部品から直接プローブされるのを防ぐために使用できます。そうした追加の対策としての効果も期待できるため、利用する価値はあります。

不揮発性の埋め込みメモリの内容は、コンポーネント内で暗号化して書き込みから保護する必要があります。マイクロコントローラとDSPの間のインターフェースは、プリント基板上のパターン・レイヤの間に埋め込む必要があります。埋め込みメモリの内容が読み取られたとしても、そのデータが暗号化されていて使用できなければ、その行為を無意味にすることができます。

メーカーは、デバッグ用のポートやテスト用のポートを挿入することがよくあります。通常はシリアル・ポートまたはJTAGポートが使われます。それらを利用すれば、システムのほとんどの部分にアクセスして制御することができます。したがって、それらのポートが、製造時に機能的に無効化または保護されていることを確認してください。デバッグ用のヘッダを装着しないというだけでは不十分です。それだけでは、悪意ある人物によって装着されたり、はんだ付けによってピンに勝手に接続が行われたりする可能性があるからです。製造中のデバイスにおいて、それらのインターフェースを有効のままにする必要がある場合には、その使用を許可する前に認証が行われるようにします。パスワードを使った保護も可能ですが、その場合はユーザが強力なパスワードを設定できるようにしなければなりません。

乱数の生成

通常、暗号化用の機能には何らかの乱数発生器（RNG：Random Number Generator）が必要になります。鍵の生成に使われる乱数は、予測が不可能なものでなければなりませんし、同じ数字が繰り返されないようになっている必要もあります。制約の多い組み込みシステムでは、リソースやエントロピが不十分であることから、乱数の生成は一般的に非常に難しい処理となります。

エントロピの乏しさは、多くの組み込みシステムに問題をもたらします。例えば、台湾で身分証として使われるID用スマート・カードについて発生したような壊滅的なハッキングにつながるおそれがあります。研究者らによると、エントロピが乏しいために、多くのIDカードで同じ数字から関連鍵が生成されていたとのことです。その結果、強力なRNGを使用していたのにもかかわらず、解読が可能になっていたと言います[9]。同様に、2012年には、公開鍵サーバ上のRSA公開鍵の0.38%で弱い乱数生成の共有が行われていたことから暗号の解読が可能だったことが、研究者らによって発見されています[10]。

RNGの強度を検証するのは困難です。というよりも、ほぼ不可能だと言えます。RNGの設計は、これまでかなり場当たり的に行われてきており、その理論はあまり確立されていませんでした。しかし近年では、堅牢な暗号用のRNGの設計とフォーマルな解析に向けた顕著な進展が見られています。

最新の堅牢なRNGは、3つのステージ構成で設計される傾向があります[8]。未処理のエントロピを提供するエントロピ・ソース、エントロピを一様に分散させるエントロピ抽出器、そして利用可能な少数のエントロピを拡大する拡大ステージから構成されます。

最初のステージは、エントロピ・ソースです。これには、クロック・ジッタや熱ノイズなど、何らかの物理的なノイズ源が使われる場合があります。アナログ・デバイセズのBlackfin® DSPなど、エントロピの生成に使用できるRNGを備えたハードウェアを提供するプロセッサも存在します。

暗号用の乱数は、統計的に一様に分布している必要があります。すべてのエントロピ・ソースにはいくらかのバイアスが存在し、暗号に使用する前にはそれを排除する必要があります。そのために使われるのがエントロピ抽出器です。エントロピ抽出器は、エントロピが高く不均一に分散した入力を受け取って、エントロピが高く一様に分散した出力を生成します。その過程で、エントロピはいくらか低下します。エントロピ抽出器は、入力以上のエントロピを備える出力を生成することはできません。したがって、暗号論的擬似乱数生成器（CSPRNG：Cryptographically Secure Pseudo-random Number Generator）のシードに使用可能なエントロピの高い小さな数字を抽出するために、それよりもはるかに多くのビットをエントロピ・ソースから収集する必要があります>[11、12]。

エラッタのエクスプロイト

ほぼすべてのIIoTノードは、何らかの組み込みファームウェアやアルゴリズムに基づいて動作します。もちろん、そのファームウェアは適切に動作するでしょうし、求められる機能を実行する能力に特に問題はないように見えるはずです。しかし、どのようなソフトウェアにも、ある程度のバグやエラッタが潜んでいるものです。セキュリティの問題につながるおそれのある異常な動作を示すこともあり得るでしょう。例えば、99.99%までエラッタを排除したファームウェアが、何らかの問題を引き起こすことはごく稀です。しかし、侵入者は、0.01%のわずかなエラッタを悪用し、特定の動作モードでノードが100%の確率で問題を起こすように細工をするかもしれません。ソフトウェアのバグは、複雑さに起因するとも言えます。しかし、複雑さは、システムが有用な処理を実行できるようにするためには避けられないものです。ソフトウェアのバグと脆弱性は、基本的にすべてのシステムに存在します。

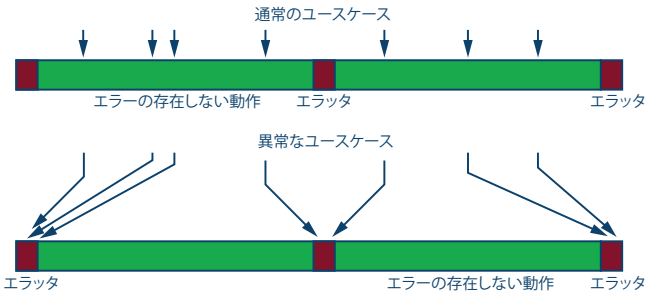

図5. エラッタの影響。小さなエラッタのエクスプロイトによって、100%の確率で障害が引き起こされるように細工されることがあります。

セキュリティを念頭に置いた設計

システムの設計は、最初からセキュリティを念頭に置いて実施する必要があります。セキュリティを確保するための仕組みは、プロジェクトの最後に追加すればよいといった類のものではありません。初期設計の時点で考慮することが必須の要素として捉える必要があります。ただ、セキュリティの確保は、そのための機能を追加するだけで実現できるものではありません。リスクの管理が必要です。あらゆるIIoTシステムの開発には、セキュリティを念頭に置いた設計手法が不可欠です。

これまで、セキュリティの確保を目的として、様々な設計手法が適用されてきました。それらはもちろん有用です。セキュリティに対する脅威をモデル化することにより、リスクを特定してそれを緩和する適切な対策を選択するとよいでしょう。システムへの侵入口を特定すれば、システムで最もリスクの高い部分を特定することができます。ほとんどの攻撃ベクトルは外部インターフェースを介したものなので、設計／実装に脆弱性がないかどうかを確認します。未知のデータは慎重に取り扱うと共に、すべての入力の検証を行います。そうした検証の対象は、侵入口だけに限定してはなりません。多層防護が重要です。つまり、外側のレイヤが破られた場合に備えて、複数のセキュリティ・レイヤを設ける必要があります。

多くのプロセッサは、複数の権限レベルに対応しています。例えば、ARM®のプロセッサにはTrustZoneが設けられています。アナログ・デバイセズのBlackfin　DSPには、ユーザ・レベルと特権レベルの実行モードが用意されています。最も低い権限レベルでできるだけ多くのコードを実行するようにし、最も重要なコードを特権モード内で保護します。IIoTデバイスのセキュリティ要件を検討する際には、セキュリティが確保できなかった場合のコスト、攻撃の可能性、主要な攻撃ベクトル、セキュリティを確保するためのソリューションの実装コストを考慮する必要があります。

まとめ

本稿で挙げた推奨事項の中には、互いに相容れないものが少なくありません。また、システムの他の設計目標とも相容れないはずのものも含まれています。セキュリティの確保を実現するには、ほぼ間違いなく何らかのトレードオフに向き合う必要があります。代償になるのは、主にコスト、機能、使い勝手です。トレードオフには、非常に効果的で代償も小さくて済むものもあれば、代償が大きいのにもかかわらず、あまり効果が得られないものもあります。セキュリティについては、設計に対する他のニーズとのバランスをとって対処する必要があります。また、セキュリティを念頭に置いた設計プロセスを採用し、アプリケーションごとに詳細を決定する必要があります。

アナログ・デバイセズは、IIoTシステムにおけるセキュリティの確保を支援するプロセッサ製品群を提供しています。それらの製品を使用すれば、ハードウェアをベースとしてセキュリティの強化を図り、エッジ・ノードの能力の限界を押し上げることができます。低消費電力のRFトランシーバー「ADF7023」は、ISM（Industry Science Medical）バンドに対応し、多くの変調方式をサポートします。また、AES（Advanced Encryption Standard）による暗号化機能を内蔵しています。

「ADuCM3029」は、トランシーバーを搭載するマイクロコントローラ製品です。AESとSHA-256の処理に対するハードウェア・アクセラレーション、真のRNG、マルチパリティ・ビットで保護されたSRAMを備えています。BlackfinファミリーのDSP「ADSP-BF70X」は、鍵を安全に保管し、セキュア・ブートを高速に実行するための組み込みOTP（One-time Programmable）メモリを備えています。それにより、感染後のシステムを既知の良好な状態に戻すことを強力に支援します。

Blackfin DSPは、ハードウェア・ベースでインクリメント専用のカウンタによるロールバック保護機能を備えています。同機能は、脆弱性が明らかになった場合にそれを修正するためのファームウェアのアップデートを可能にします。この機能とデータの変更が禁止された鍵用のストレージによって、堅牢でレジリエントなエッジ・ノードを構築することができます。また、Blackfin DSPは、暗号処理用のハードウェア・アクセラレータ、ハードウェア・ベースの真のRNG、特権レベルと非特権レベルのコードの実行を分離する機能、MMU（メモリ管理ユニット）を備えています。多くのDMA（Direct Memory Access）チャンネルへのアクセスを制限することが可能なので、電力効率の高いセキュアなデジタル信号処理を低コストで並列実行することができます。

参考資料

[1] Mirai botnet leaked source code（Miraiのボットネットによりリークされたソース・コード）

[2] Ross Yu「Security and Reliability Are Key in Wireless Networks for Industrial IoT（IIoT用ワイヤレス・ネットワークの鍵を握るセキュリティと信頼性）」Analog Devices、2017年.

[3] Patrick Nelson「Organizations Must Isolate IoT from Regular IT, Says Telco（「IoTは、一般的なITから隔離すべき」――通信会社が指摘）」Network World、2016年３月

[4] Brian Girardi「Endpoint Security and the Internet of Things（エンドポイントのセキュリティとIoT）」CSO、2017年

[5] Tristan O'Gorman「A Primer on IoT Security Risks（IoTにおけるセキュリティのリスク）」Security Intelligence、February 2017年２月

[6] Abhijeet Rane「IoT Security Starts with Secure Boot（セキュア・ブートで始まるIoTのセキュリティ）」Embedded Computing Design、2017年１月

[7] Amitrajan Gantait、Ayan Mukherjee、Joy Parta「Securing IoT Devices and Gateways（IoTデバイスとゲートウェイの保護）」IBM、2016年５月

[8] Boaz Barak、Shai Halevi「A Model and Architecture for Pseudo-Random Generation with Applications to /dev/random（疑似乱数生成のモデルとアーキテクチャ、/dev/randomへの応用）」Proceedings of the 12th ACM conference on Computer and communications security, November 2005年11月

[9] Chen-Mou Chang、Daniel J. Bernstein、Chang、Li-Ping Chou、Nadia Heninger、Nicko van Sormersen、Tanja Lange、Yun-An Chang「Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild（認証済みスマート・カードから取得したRSA鍵の因数分解――野放しの銅細工師）」Springer、2013年

[10] Arjen K. Lenstra、Christopher Wachter、James P. Hughes、Joppe W. Bos、Maxine Augier、Thorsten Kliengun「Ron Was Wrong. Whit Is Right（Ronは誤っていた。正しいのはWhitだ）」Cryptology ePrint Archive、Report 2012/064、2012年

[11] Boaz Barak、Eran Tromer、Ronen Shaltiel「True Random Number Generators Secure in a Changing Environment（変化する環境において安全性を維持するための真のRNG）」Springer、2003年

[12] Boaz Barak、François-Xavier Standaert、Hugo Krawczyk、Krzysztof Pietrzak、Olivier Pereira、Yevgeniy Dodis、Yu Yu「Leftover Hash Lemma, Revisited（残余ハッシュ補助定理の再考）」31st Annual Conference on Advances in Cryptology、2011年８月

著者について

Ian Beavers（Ian.Beavers@analog.com）は、アナログ・デバイセズのオートメーション／エネルギー／センサー・チーム（ノースカロライナ州グリーンズボロ）を担当する製品エンジニアリング・マネージャです。1999年に入社以来、19年間にわたって半導体業界で経験を積んでいます。ノースカロライナ州立大学で電気工学の学士号、ノースカロライナ大学 グリーンズボロ校で経営学の修士号を取得しています。

Erik MacLean（erik.maclean@analog.com）は、アナログ・デバイセズのセキュリティ・ソリューション・グループ（フロリダ州タンパ）に所属するハードウェア技術者です。組み込みハードウェアの設計とセキュリティに関する業務に6年間従事してきました。2009年にフロリダ大学で電気工学の学士号を、2011年にサウスフロリダ大学で電気工学の修士号を取得しています。

オンライン・サポート・コミュニティ

EngineerZone™ SUPPORT COMMUNITY

アナログ・デバイセズのオンライン・サポート・コミュニティに参加すれば、各種の分野を専門とする技術者との連携を図ることができます。難易度の高い設計上の問題について問い合わせを行ったり、FAQ を参照したり、ディスカッションに参加したりすることが可能です。

ez.analog.com にアクセス

＊英語版技術記事はこちらよりご覧いただけます。

アナログ・デバイセズ株式会社

本　　社 〒105-6891　東京都港区海岸1-16-1　ニューピア竹芝サウスタワービル10F
大阪営業所 〒532-0003　大阪府大阪市淀川区宮原3-5-36　新大阪トラストタワー10F
名古屋営業所 〒451-6040　愛知県名古屋市西区牛島町6-1　名古屋ルーセントタワー40F

TA16680-0-3/18

www.analog.com/jp

ANALOG DEVICES
想像を超える可能性を
AHEAD OF WHAT'S POSSIBLE™