M2M 通信を行う IoT デバイスの

デバッグポートと実時間性に対するセキュリティ脅威

情報セキュリティ大学院大学

松井俊浩*

概 要

2010 年代から、あらゆるモノがインターネットに接続される IoT と呼ばれるビジョンが広まっている。IoT では、デバイスがインターネットに接続されるために、デバイスをインターネットから操作されたり、インターネットから情報をのぞき見られることが心配されているが、IoT デバイスが攻撃者の手の届く場所にあり、ユーザーの介入のない M2M 通信を行うため、より直接的な攻撃を受けやすいことがより重大な問題である。また、資源制約が強いことと実世界を制御することから、実時間制御への攻撃などの新手の問題が生じてくると想定される。これらの IoT デバイスの特質と、特徴的な攻撃として、デバッグポートと実時間性に対する攻撃について考察する。

1 IoT を巡る状況

IoT、すなわち Internet of Things という用語は、1999 年に Kevin Ashton が Procter & Gamble の講演会で初めて使用したとされる。Ashton は、我々の生活や社会は、モノに大きく依存しているのであるから、コンピュータは、自分自身で世界のモノを見たり聞いたりすることができるべきであり、そのために RFID やセンサーが活用されるべきであろうと述べた[1]。その後、IoT の概念は、RFID などから離れて、拡張してきた。2010年頃には、並行して CPS (Cyber Physical System)が提案された。CPS は、世界がクラウドで制御されるロボットやセンサーネットワークになるというアイデアであり、生活や社会の中に IT や組み込みシステムが浸透するという意味では、IoT と同義と言って良い。生活や社会にコンピュータが普及していく姿は、1990年頃にユビキタスコンピューティングとして予想されていた。IT が浸透する現象を捉えて、Pervasive Computing と呼ばれることもあった。2000年に総務省が推進したユビキタスネットワークは、どこでもブロードバンドなイ

* 情報セキュリティ研究科 教授

ンターネットに接続できることを標榜した言葉で、IoT とは異なるが、世の中のさまざまな機器が IT 化されることは、かなり以前から予想されていたことは間違いない。

根底には、Moore の法則によって予想される、機器の小型化、高密度化があった。Moore の法則によれば、10 年で 100 倍、20 年で 1 万倍、30 年で 100 万倍、40 年で 1 億倍の高密度化が達成される。Moore 則に後押しされて、電話やカメラのような小型機器でも 1Gips 以上の計算が可能になった。さらに、2000年の後に生じた 802.11 系の LAN ワイヤレス通信および 3G, 4G と続く公衆ワイヤレス通信の爆発的な普及と高速化によって、IoT が現実の物となってきた。マイクロデバイスとワイヤレス通信という、もっぱらハードウェア的進化が、IoT を可能にしている。

IDC は、IoT を次のように定義している：

> *「A network of networks of uniquely identifiable endpoints (or "things") that communicate without human interaction using IP connectivity (IP接続による通信を、人の介在なしにローカルまたはグローバルに行う個体識別可能な端点(モノ)からなるネットワークのネットワーク)」*[3]。

注目すべきは、人の介在がないという部分と、network of networks というフレーズである。人の介在のない通信は、いわゆる M2M 通信を意味する。だから、たとえば、人がブラウザを使って情報検索をするような場面は、IoT とは関係がない。a network of networks とは、LAN のようなネットワークをつないだ大規模なネットワークが想定されるが、ネットワークは規模によらずネットワークであることには違いがない。類義語である system of systems(SoS)は、統御者のある独立なシステムをさらにつないだ大規模システムで、全体の統御者がいないシステムを a system of systems としている。したがって、大規模なネットワークには統御者がいるが、a network of networks には、全体の統御者がいないということになる。したがって、モノが IP アドレスを持ってネットワークにつながるが、デバイスが M2M 通信を行い、ネットワーク全体の統御者がいないということが、IoT の特質と言うことになる。

さらに、ITU-T は、2012 年に、モノには仮想的なモノも含まれるとしている[4]。CISCO は、2013 年に、モノの他に、人々、プロセス、データがインターネット化される社会を想定して、IoE (Internet of Everything) を提唱した[5]。IoT が、モノに注目しているのに対し、これらの拡張概念は、これまで発展してきた情報系あるいはサイバー空間と IoT の連携を見ている。サイバー空間での脅威が、モノの世界にも波及する、モノとサイバー空間の間で脅威が広がることも暗示している。

2 サイバーセキュリティにとっての IoT の特質.

前節で、IoT の概念や定義を述べたが、IoT は、標準や規格ではなく、ビジョンに過ぎないから、必ず IoT 社会にならなければならないわけではない。セキュリティの観点からは、今後、どのような脅威が新たに生じてくるかを、IoT というビジョンや傾向から読み取りたいのであるから、IoT の定義に外れた機器に対して、IoT 的でないなどと難癖を付けるべきものではなく、そのような機器がたくさん生じてくるなら、そこでのセキュリティを検討する必要がある。繰り返すが、IoT の定義が重要なのではなく、IoT のようなビジョンが世界に広がることで新

たに生じてくるセキュリティの脅威が問題なのである。そこで、サイバーセキュリティにとっての IoT の特質を以下の 5 つに分けて考える†。そのセキュリティ脅威や対策は後述する。

① M2M: 人が介在しない、装置同士の接続が行われる
② 実世界性: 実世界の物理的実体から情報を収集する、または制御すること
③ ロングライフ: 長期間運用されること
④ 資源制約: 小型で資源制約が強いこと
⑤ 多数性: 低コストで多数が普及すること

2.1 M2M 通信

現在までのインターネットは、パーソナルデバイス(クライアントとなる PC,スマートフォン、タブレットなど)が、サーバーにアクセスして、Web サイトを閲覧したり、流されている動画を見たり、Email を交換することに使われてきた。インターネットには、多数のパーソナルデバイスが接続されているが、実際にはパーソナルデバイス同士が互いに通信することは全くなく、通信は、パーソナルデバイスとサーバーの間に限定されていた。ここで、パーソナルデバイスとは、ユーザー、すなわち人が使うのであり、したがって、表示用の画面やポインティングデバイスを備えている。

IoT におけるデバイスから生じる情報は、監視カメラから映像記録器、スマートメータから電力量集計センター、自動車内の燃料噴射器からエンジン制御器、ETC カードと ITS の間などのように、人を介さない接続を通じて交わされるのが特徴である。インターネットにおける情報交換は、人間との情報交換が主目的であり、その安全性の確認には、パスワードのような人の目を使うことが出来た。これは、トラストの基点を、コンピュータの中の情報ではなく、ネットワークからは直接の操作が不可能な人間の脳の中に置くことに相当する。M2M では、人間が介在しないので、人間の脳というオフラインで侵されることのない場所に置かれたパスワードを使用することができない。パスワードを暗号化するなどして機器のメモリに書き込んでおくことは出来るが、何らかの方法で読み取られ、改変される危険性を排除できない。

IoT の M2M 通信では、パスワードを用いた認証が使えなくなるだけでなく、人間による日頃の監視が困難になる。危険な操作を行うごとに管理者の判断を求めるアラートを出すことはできないし、管理者は、動作が遅いから走っているタスクの状況をモニタすべき局面があっても、動作が遅いと言うことに気づくことが出来ない。エッジデバイスで収集されるデータは、インターネット接続を通じて、クラウドに送り込まれるが、ユーザーは、そのようなデータ伝送に気づくことも少ない。

2.2 実世界性

従来のインターネットが、インターネットだけで閉じたサイバー空間（情報世界）を構成していたのに対し、CPS とも呼ばれる IoT は、実世界の物理的実体から情報を収集し、実世

†総務省の IoT セキュリティガイドラインでは、次の 6 つの性質を IoT セキュリティの特質としている: ①脅威の影響範囲/影響度合いが大きいこと、②IoT 機器のライフサイクルが長いこと、③IoT 機器に対する監視が行き届きにくいこと、④IoT 機器側とネットワーク側の環境や特性の相互理解が不十分であること、⑤IoT 機器の機能/性能が限られていること、⑥開発者が想定していなかった接続が行われる可能性があること。

界に働きかけて、物理的実体を制御する。すなわち、モノを動かしたり、熱したりする作用を行う。したがって、IoT の情報系のセキュリティが、安全性(セーフティ)に直結する。

実世界を扱うために、機器が物理的に露出する。その場所も、住宅やオフィスのような保護された環境ではなく、公共の場や野外に露出することがある。これによって、まず機器が破壊、盗難、改ざんされる危険性が高まる。外部メモリとしてメモリカードなどを使っていれば、メモリカードの内容をコピーされても気づかないことが起こり得る。

また、機器やネットワーク配線に信号線を直結しての侵入が簡単になる。無線ネットワークは、四方八方に電波をまき散らすが、有線は漏洩がないので安全と考えられやすい。しかし、有線ネットワークは、そこに情報が流れていることが明らかにわかるし、物理的に露出しているのであれば、電波を傍受するのも、ネットワークケーブルにタップを立てるのも、苦労は変わらない。実際は、無線 LAN は厳重な暗号化がされており、有線 LAN は平文を流すことが多いので、無線 LAN の方が安全性が高い。

特に抜け穴となりやすいのが、PCB(プリント配線板)にメインテナンスのために用意されたテストパッドや JTAG ポートである。これらの開発・デバッグ用のポートには、マイクロコントローラ内のメモリ内容を読み出したり書き込むコマンドが用意されている。従来、これらは外部に露出しないことを前提にされていたが、ネットワークからのサイバー攻撃よりはるかに簡単で強力な侵入法となる。最近は、USB ポートの危険性は良く認識されてきたが、メインテナンス用のテストパッド、JTAG ポートは、一般人には存在を知られておらず、簡単にどこにあるか見えないだけに盲点となりやすい。

保守ポートだけでなく、クロック線、電源線などもシステムに擾乱を与える入り口となる。クロック線からは、正規のクロックに対してグリッチ(ひげ)を追加するクロック・グリッチ攻撃が成立しうる。暗号鍵の正否を検査する分岐命令にグリッチをぶつけると、分岐命令が異常動作を起こして、間違った暗号鍵を正しいと認識させることが可能になる[12]。電力線のモニタして、マイクロコントローラが消費する電力を精密に計測されると、やはり暗号鍵を抽出されるサイドチャネル攻撃が成立しうる。

実世界を操作する IoT 機器は、深刻な個人情報を扱う例もあるが、機密性のレベルは総じて低い。ペースメーカや糖尿病治療のためのインシュリンポンプなどの医療機器、工場の制御システム、自動車のコントローラなどでは、内部を行き交う情報の機密性よりも、物理的動作を確実に行う可用性と完全性が重要である。その影響は、健康、人命、災害などにも関わり、多大な被害を与えうる。

2.3 ロングライフ

パソコンやサーバーの法定減価償却期間は、4-5 年であるが、PC の買い替え期間は、内閣府の消費動向調査によると 2016 年では平均 6 年である。IoT 機器の寿命はそれより長く、たとえば、自動車の耐用年数は 4-6 年であるが、実際の平均使用年数は、一般財団法人自動車検査登録情報協会によると、12-13 年である。オフィスソフト、サーバーの基本ソフトやアプリケーションは、1 年と同じ状態が続かず、新しいソフトウェアに乗り換えたり、アップデートを受けたりする。IoT 機器の組込ソフトウェアも、アップデートによって改良されていくであろう。たとえば、自動車に何らかの欠陥が見つかってリコールを受ける場合、車載 ECU のソフトウェアがアップデートを受ける機会が増えるであろう。

ソフトウェアアップデートは、前節でも述べたように、マルウェアを機器内に招き入れてしまう機会となり得る。アップデートを通信によって、ダウンロード元の証明書を確認しながら行う場合は、危険性は少ないが、たとえば、自動車が修理工場で修理工が正しいと信じているアップデートをオフラインで施すような場合は、アップデートプログラムをすり替えられる危険性が増大する。

2.4 資源制約

PC のユーザーは、多かれ少なかれ、自らの PC が高性能であることを望んでいるが、IoT 機器では、正反対に、なるべく低性能のプロセッサが選択される。たとえば、自動車の ECU に搭載されるマイクロコントローラは、コスト削減のために必要な機能、実時間性能を達成するためのぎりぎりの性能、メモリ容量のプロセッサが選択される[15]。冗長構成が取られることも少ない。このため、セキュリティのための暗号化や認証を実行するための余分の計算資源が得られるとは限らない。もちろん、セキュリティリスクを減らすためにより高性能のマイクロコントローラを選択すべきことはわかっているが、PC のようにセキュリティがユーザーにとっても切実な価値であることが、IoT においてはわかりにくく、コスト上昇の理解を得にくい。

2.5 多数性 (Multiplicity)

IoT 機器は、車載、家電、スマートメータなど、多数が世に出る。ほぼ均質な構成の機器が多数出回るので、もし脆弱性が含まれていれば、多数が同様な攻撃の対象になり得る。そして既出の M2M の特性によって、ユーザーあるいは管理者の知らないところで被害が蔓延する可能性がある。たとえば、多数の監視カメラが、DOS 攻撃に駆り出されるような事態が生じやすい。

3 IoT のセキュリティ脅威

2 節で述べた IoT の特質に基づき、IoT において、新たに発生しうる脅威を予測する。IoT では、これまで独立していたモノが、インターネットに接続されるのであるから、インターネットから不正アクセスやマルウェアの感染に注意すべきであると説明されることが多い[2, 6, 7, 8]。たとえば、つながる世界のセーフティ＆セキュリティ設計入門[7]では、大学等のコピー複合機が意図せずにインターネットに接続され、デフォルトのままになっていたパスワードによって、外部から不正アクセスを受ける状態であったことや、上位のネットワークからマルウェアに感染する事例が取り上げられている。制御システムネットワークのセキュリティについても、これまでインターネットに接続していなかった機器が、IP 接続を通じて保守を受けるようになって改めてセキュリティ脆弱性や設計法、管理体制が問われるようになった。このような、新たなインターネット接続が引き起こすセキュリティ脅威は、非常に重要であるが、制御システムにおいて明らかにされた、機器の EDSA 認証を受ける、ホワイトリスト制御を行う、故障とセキュリティインシデントを識別するなどの知見を生かすことで対応が可能である。より IoT デバイスに特徴的な、M2M 通信や物理的に直接攻撃を受けやすい脅威について論ずる。

3.1 M2M 通信の偽装

ユーザーの監視下で行われるパーソナルデバイスとサーバーの間で秘匿したい通信を行う場合、利用者の ID とパスワード、あるいは指紋や静脈などのバイオメトリクスを用いた認証が行われるのが普通である。認証に必要な情報をパーソナルデバイスの外、すなわちユーザーの脳の中や指先の指紋に置くことで、パーソナルデバイスを盗難から守っている。しかし、実際は、ブラウザ等が ID やパスワードを記憶するので、パーソナルデバイスのどこかに、認証に必要な情報が格納されている。それでも大きな問題とならないのは、そのパーソナルデバイスを使用するために Windows や Linux へのログイン認証が必要で、このログイン認証には機械に記憶させた情報が使えず、毎回ユーザーが入力しなければならないからである。

同様の認証方法を IoT デバイスで考えてみる。IoT 機器が M2M 通信を行うといっても、多くの場合、その IoT 機器の持ち主や、そのデバイスが収集する情報の使用者は特定できる。簡単には、その装置の電源を入れて使用を始める人がいれば、その人が上記の Windows 等のログインユーザーに相当する。たとえば、自動車であれば、使用時にパスワード認証は行わないが、その自動車のキーを持っている人がログインユーザーであり、キーを回す(あるいはキーを持ってイグニッションボタンを押す)操作がログイン操作に相当する。ログインに成功すれば、機器内に格納された各種の暗号鍵をアクティベートして、その後の自動車内の M2M の認証を連鎖させていくことが出来る。

ここでの問題は、やはり、犯罪者が、IoT 機器に直接、物理的に接触できることがあるため、デバイスに保存された各種暗号鍵を窃取される可能性があること、さらに、キーがなくてもエンジンを始動するような細工も可能なことである。

未来の自動車は、自動運転を円滑に行うため、安全確保のために車々間通信や路車間通信を盛んに行うことが予想される。たとえば、交差点の手前で、交差点に進入しようとする自動車同士が位置や速度を交わせば衝突回避の効率が良くなる。この通信には、人は介入しない。衝突回避モジュールが、ワイアレス通信で付近の自動車や信号機などを呼び出す M2M 通信となる。ここで通信相手を偽装して誤った情報を挿入することで衝突を誘発することができる。通信相手が正当な自動車等であることを保証することは、非常に困難である。

3.2 不揮発メモリ

現在、コンピュータで主に用いられるメモリは、フリップフロップで構成される、高速性に優れた SRAM と、セルの専有面積が小さいので大容量で高速のメモリが構成できる DRAM である。SRAM が記憶内容を保持するには、読み書きを行わない期間にも継続的な電力供給が必要である。DRAM は、小さなキャパシタの静電容量を長期間保持するために、各セル当たり 64ms 周期でのリフレッシュ動作を必要とする。いずれも、記憶保持に電力が必要であることを意味している。裏返すと、現行のすべての情報システムは、電源が失われると主記憶の内容が失われることを前提に設計されている。

PC は、盗難されたとしても、電源が切られていれば、メモリ中には情報は残っておらず、情報を摂取しようとすれば、ディスクから暗号化された文書等の復号化を試みるしかない。ところが、メモリ中には、攻撃者にとって、うまくすれば、平文化された情報や、暗号鍵が残

っている可能性がある。実際、DRAM は、急冷すれば記憶内容を長時間保持できる性質があり、この特性をついたコールドブート攻撃が成功することが示されてきている[13]。また、AndersonとKuhnは、ATMなどに用いられているSRAMでは、非常な長期間同じ内容を保持しているので、電源断の後の再起動によって、高確率で同じ内容がよみがえることを指摘している[9]。IoT デバイスは、2.3 節で述べたように、長期間同じ環境で同じ動作を継続することがある。そこで、メモリのある決まった番地に暗号鍵が保存されているような状況では、電源を切っても暗号鍵の残像が保存され、攻撃者によってデバイスが持ち去られた後に暗号鍵が盗用される危険性が残る。

NEDO が2016年から実施している IoT 横断技術開発プロジェクトでは、IoT に必要な要素技術開発として、センサー系、ストレージ系、人工知能系、セキュリティの4つを定めている。この IoT 向けストレージとしては、省電力特性に優れた不揮発メモリの開発が行われる。不揮発メモリは、IoT デバイスのような節電性能要求の強い機器には、効果的なメモリデバイスであり、PC 等に先駆けて採用される可能性が高い。そして、このような IoT デバイスでは、冷凍保存のような特別な処理を行わなくとも、主メモリの秘密情報を簡単に読み取られてしまう可能性が生じる。

3.3 実時間制御

実世界を対象とするIoT機器は、物理法則に従い、一定の周期でデジタル制御を行う、実時間制御を特徴とする。たとえば、UAV が、安定に飛行するためには、ミリ秒周期でのセンサーの読み取りとアクチュエータの駆動を行う必要がある。そして、2.4 で述べたような資源制約により、内蔵されるコンピュータシステムの能力には、余裕が少ない。この状態で、マルウェア等が新たな計算負荷を与えると、それだけで制御が不安定になる可能性がある。もちろん、制御タスクを実行させる実時間カーネルは、優先度に基づく実時間スケジューリングを行うが、実時間スケジューリングは、EDF アルゴリズムも、RMS アルゴリズムも CPU の負荷率に敏感であるから、ぎりぎりの計算能力しか与えられていないデバイスでは、わずかな負荷の増加が深刻な結果を生みうる。たとえば、マルウェアがキャッシュメモリを汚してしまうだけで、制御のタイミングは狂う。特別に複雑な攻撃を加えずとも、広い範囲のメモリに書き込むような簡単なプログラムでも、UAV を墜落させてしまうかも知れない。

同様な、負荷を与えるだけの攻撃は、CPU がスリープモードに入れなくして、省電力性を阻害する攻撃にもなり得る。

3.4 リバースエンジニアリング

IoT デバイスは、データセンターのサーバーと異なり、自動車にしても医療機器にしても、物理的に人が触れる場所に設置されるので、ISMS で言うところの物理的攻撃を受けやすい。物理的攻撃には、機器の破壊、窃取、メモリカードの抜き取りのような、あからさまな攻撃もあるが、機器を制御する末端のデバイスには、Email が届くわけでも、装置の設計情報が隠されているわけでもなく、それによって窃取される情報自体の被害は小さいと思われる。深刻なのは、メモリ中のプログラムやパラメータを書き換えて戻されることである。すなわち、デバイスは、インターネットとは通信をしないと信じていても USB メモリからマルウェアが注入されるのと同様、開発・保守ポートから悪性プログラムが注入される。USB メモリ等からの

マルウェアは、デバイスの制御プログラムに何らかの脆弱性を利用しない限り、システムの誤動作を引き起こすことは難しいが、開発・保守ポートからの侵入では、デバッグコマンドを使って、マイクロコントローラ内の ROM、RAM の内容を読み出し、逆コンパイル等のリバースエンジニアリングによって書き換え、書き戻すことが可能である。オペレーティングシステムやデバイスドライバを書き換えてしまうので、特定の脆弱性に頼る必要がなく、直接に誤動作プログラムを作り込める。このような攻撃をここでは*直接改変攻撃*と呼ぶ。ISMS では、技術的脅威、人的脅威、物理脅威の三つの脅威に分類しているが、デバッグポートからの直接改変攻撃は、どれにも属さないように考えられる。

たとえば、車載エレクトロニクスに対しては、修理工場がこのような物理攻撃を行うのに好適な場所となるであろう。情報システムでは、重要情報が格納され、高可用性が求められるサーバーは、サーバー室自体に施錠や監視カメラなどの物理セキュリティが求められる。これからは、修理工場が車載ネットワークへの侵入現場であるとの認識を広めて、IoT 機器が露出する場所での物理セキュリティを高める必要がある。

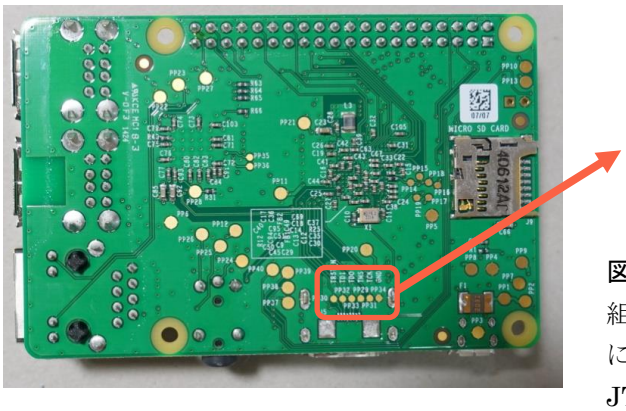

図 1.　組込システムの開発・保守ポート
組込学習ボード Raspberry Pi のボード裏面にあるテストパッド。TRST, TDI, TDO などは、JTAG の信号名。

3.5　クラウドへの情報集約

IoT デバイスは、機器の使用に伴うデータを順次、クラウドサーバに送る。たとえば、ヘルスケアや自動車等のメインテナンスサービスのために、人の行動や自動車の稼働状況に関する情報の把握が必要だからである。また、監視カメラの映像をクラウドに集め、AI を使って集中的かつ自動的に監視業務を行うサービスなども考えられる。この場合、情報が集まることによって AI の学習をさらに進めることができるので、IoT によってサービスの質が高まっていくことが期待できる。これらの情報は、基本的に個人情報であるので、暗号化等適切な方法で通信が行われると推定されるが、これまで述べてきたように、IoT デバイスに直接的に侵入されてプログラムを改変等された場合、暗号化される前の情報が抜き取られ、インターネットを通じて犯罪者のサーバーに送られる被害が出るかも知れない。

4　対策

前節までで、機器に対する物理的アクセスとプログラム改変、また M2M 通信という IoT

デバイスの特質によって生じうるセキュリティ問題を論じた。これらに対抗していくために、以下のような備えが必要と考えられる。

4.1 開発･デバッグポートからのプログラム改竄

4.1.1 開発/デバッグポートを閉じる

主たる問題は、開発・デバッグポートから IoT デバイスのプログラムを書き換えられてしまうことに起因する。開発・デバッグポートは、メインテナンスにとって重要であるが、開発やメインテナンスでは、システムのあらゆる資源にアクセス可能でなければならないので、大きなセキュリティの綻びにもなる。両者を比較して、脆弱性の問題の方が大きいのであれば、大量に出荷される製品については開発・デバッグポートを設けない、塞いでしまうべきである。

しかし、組込まれるマイクロコントローラ LSI 自体は、デバッグポートを持つことは間違いないので、LSI のピンに直接接続されれば、同じことになる。その場合でも、LSI のピンは、数種類の機能がオーバロードされるのが普通であるので、初期状態でデバッグポートモードにならないようにしておくべきである。これによって、何らかのプログラムを走らせないとデバッグモードに入れないので、そのプログラムが、正当なデバッグなのか、攻撃なのかを識別するチャンスを与えることができる。また、デバッグモードに入れるコマンドを十分に管理すべきである。

4.1.2 開発・デバッグポートへの物理的アクセスを保護する

ここで論じている問題は、無防備な IoT デバイスの開発・デバッグポートが、物理的にむき出しになっているので、組込ソフトウェアを改ざんする攻撃が簡単に成立してしまうことである。これを防ぐには、ハードウェアにハードウェア的な耐タンパー性を持たせる必要がある[10]。

耐タンパーには、タンパーを検出して抑止する考え方(tamper detection) と、タンパーされたことの証跡を残すことで、タンパーしにくくする方法 (tamper evidence) がある。後者が成立するためには、タンパーされた機器を事後に回収し、解析できなければならない。しかし、データセンターのサーバー機などと異なり、IoT デバイスは犯人が持ち去られることも多い。たとえば、車載エレクトロニクスへの攻撃が成功すれば、自動車もろとも盗難されてしまう。すなわち、タンパーと同時に盗難されてしまうような機器では、タンパーを検出して抑制する方式の要求が高い。図は、タンパーによってデバイスが正常に動作しなくなるような仕組みの一例である。

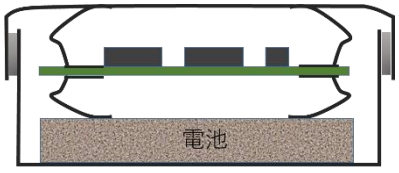

図 2　耐タンパー型の IoT デバイス
蓋をこじ開けてアクセスしようとすれば、電池からの電力が絶たれてメモリ内容が失われる[7]

4.1.3 書き換え不可能な ROM プログラムによる改竄防止

三つ目の方法は、プログラムの改変を検出することである。しかし、この検出プログラム自体が書き換えられるのでは議論が成立しない。したがって、プログラムの改変検出プログラムは、決して書き換えられない ROM に置かれるべきである。この ROM は、電気的に書き換え可能な Flash ROM ではなく、製造時にパターンを焼き込まれるマスク ROM のことである。フューズ型の ROM であってもレーザーによってフューズを溶断し、書き換えることが不可能ではない。ROM をトラストの基点とすることになる。改変の検出には、たとえばプログラム全体のハッシュを計算して、信頼できるサーバーに送って検査を受ける。

この ROM チップをマイクロコントローラとは別のチップにすると、チップの置き換えによるプログラムの改変を許してしまう。CPU と同じチップに作り込まれているべきである。そのような構成は、マイクロコントローラの自由度を失わせるし、専用チップを作ることになるのでコストアップにつながる。

4.2 実時間性を守る

実時間カーネルは、プロセスの優先度に応じて計算時間を割り当てる。優先度の割り当ては、プログラムが指定する場合もあるが、カーネルが自動的に割り当てる場合は、EDF あるいは RMS アルゴリズムに従う場合が多い。EDF (Earliest Deadline First)では、処理を完了すべき〆切時刻を指定する。RMS(Rate Monotonic System)では、実行周期を指定する。各プロセスは、スケジューリングのパラメータとして、優先度、〆切時刻、実行周期を記録している。したがって、EDF において、マルウェアが〆切時刻として直近の時刻を指定すれば、他のプロセスに優先してマルウェアが実行されることになる。RMS においては、ごく短い周期を指定すれば CPU 時間を奪うことができる。実時間カーネルは、各プロセスがこれらのパラメータを正直に申告することを期待しており、正常なプログラムセットでは、CPU に無駄をさせず良い性能を出すためにすべてのプロセスが協力し合うことを前提としている。そのような環境では、マルウェアが実時間カーネルをだますことはたやすい。そのような、協力し合う体制においても、実際のプロセスの処理に推定値以上の時間がかかったり、実行周期が揺れることは起こりうるので、申告値を違えて CPU を使うプロセスを停止させることは、必ずしも正しくはない。

すなわち、複数のプロセスが協力し合って実時間制約を守ろうとしているが、たまに制約が守れないことも許容しなければならないとすると、実行プロファイルからマルウェアを特定することは極めて難しい。この中でマルウェアによる実時間性攻撃を防ごうとするなら、実行に入ることができるプロセスをあらかじめ用意したホワイトリストで制限することしかない。キャッシュを汚す攻撃も、同様にそのようなマルウェアを実行時に検出したり停止させることは困難であろう。ホワイトリストを利用することが最も現実的であるが、プロセスがすべてのキャッシュを使い切ることがないようにする MMU の機能を使用することも有効である。

4.3 暗号処理

暗号は、一般には機密性保持のために用いられるが、IoT デバイスでは、改ざん防止、すなわち完全性の保持のために重要となる。IoT デバイスのマイクロコントローラには、暗

号処理も行えるように性能に余裕を持たせた設計をする必要がある。この余裕は、前節の実時間性への攻撃にとっても有効である。暗号処理は、次節で述べるデバイスの認証にとっても必要である。

より完全なセキュリティを求めるならば、メモリ中のプログラムを暗号化しておき、実行時に解読して使うのが良い。実行の前に平文に解読するの方法でも効果はあるが、命令サイクルごとに解読する方が安全性は高い。橋本らは、そのようなセキュリティプロセッサを提案している[11, 14]。暗号鍵の配送方法が問題となる。

4.4 IoT デバイスの認証

2.1 節では、IoT デバイスは、ユーザーがログインするわけではなく、M2M 通信を行うので、ID とパスワードによる認証が困難であることを述べた。ある IoT デバイスに接続してくる別のデバイスが、接続権限を持った正当なデバイスであることを、チャレンジレスポンス方式などで確認したいのだが、そのデバイスに隠された暗号鍵は、開発/デバッグポートへのアクセスによって攻撃者に知られてしまっている可能性がある。レスポンスを生成する関数が、デバイス内のメモリ内容やそのハッシュ値で変化するような方式が考えられる。

あるいは、暗号鍵やメモリ内容が改ざんされていないことをブロックチェーンで確認する方法がとれるかもしれないが、現在のブロックチェーンはトランザクションに時間を要するので、IoT のような大量のデバイス間の通信に適用することは困難かもしれない。

4.5 プログラムのサーバー供給

IoT デバイスの Flash メモリなどに格納されたプログラムが改変される怖れがあるのなら、使用の都度、サーバーからダウンロードすることで改変を回避できる。サーバーと IoT デバイスの間にセキュアな通信路を確保できるのであれば、無防備なメモリを攻撃者の手の届く場所に置いておくよりセキュリティを高くできる。問題は、セキュアな通信のために認証あるいは証明書が必要で、それらは書き換えられる可能性があることである。

5 終わりに

IoT デバイスに対するセキュリティ脅威について、デバッグポートと実時間性に対する攻撃を中心に論じた。IoT は、デバイスがインターネットに接続されて、インターネットからマルウェア等が侵入することが強調されるが、インターネットへの接続は、PC クライアントあるいは PC サーバーが接続するのと同様の対策を施せばよい。より申告なのは、これらのデバイスが、物理的に攻撃者の手の届く場所に置かれることである。それによって、開発・デバッグ用ポートから容易に侵入され、メモリ内容を改ざんされる。これを防ぐには、IoT デバイスの対タンパー性を高める必要があることを述べた。また、資源制約が強いのでプロセッサは十分な性能を持たず、したがって実時間性への攻撃にも弱いことを述べた。実時間システムは、そもそもプロセス同士が協調し合って実時間性を発揮することを期待しているので、少しでも性能を落とす行動を取るプロセスがあると全体が破綻しうる。ホワイトリストなどによる制御が必要となる。

すべてを事前に想定することは困難であるが、IoT のような新たな局面が生ずる場合、あらかじめリスクを予測することは、リスク分析の基本である。想定外の事態が起こってあわてないよう、さまざまな事態を広く予測すべきである。

参考文献

[1] Kevin Ashton, "That 'Internet of Things' Thing", RFID Journal, 22 June 2009.
[2] —-, ”安全な IoT システムのためのセキュリティに関する一般的枠組”、 内閣サイバーセキュリティセンター、2016 年 8 月。
[3] —, “Explosive Internet of Things Spending to Reach $1.7 Trillion in 2020, According to IDC,” http://www.idc.com/getdoc.jsp?containerId=prUS25658015, IDC Press Release, 2015.
[4] ITU-T, “Overview of th Internet of things,” Y.2060, Telecommunication Starndization Sector of ITU, 2012.
[5] —, “The Internet of Everything,” http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy_FAQ.pdf, 2013.
[6] —、“IoT セキュリティガイドライン”、総務省、http://www.soumu.go.jp/menu_news/s-news/01ryutsu03_02000108.html、 2016 年 7 月。
[7] —、”つながる世界のセーフティ&セキュリティ設計入門 ～IoT 時代のシステム開発『見える化』～”、独立行政法人情報処理推進機構（IPA） 技術本部 ソフトウェア高信頼化センター（SEC）、2015 年 10 月.
[8] —、”つながる世界の開発指針 ～安全安心な IoT の実現に向けて開発者に意識してほしい重要ポイント～”、https://www.ipa.go.jp/files/000054906.pdf、独立行政法人情報処理推進機構（IPA） 技術本部 ソフトウェア高信頼化センター（SEC）、2016 年 7 月.
[9] Ross Anderson and Markus Kuhn, “Low Cost Attacks on Tamper Resistant Devices,” M Lomas et al. (ed.), Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings, Springer LNCS 1361, pp 125-136, ISBN 3-540-64040-1.
[10] Oliver K ömmerling and Markus G. Kuhn, “Design Principles for Tamper-Resistant Smartcard Processors,” Proc. USENIX Workshop on Smartcard Technology, pp. 9-20, 1999.
[11] Sean W. Smith, Steve Weingart. Building a High-Performance, Programmable Secure Coprocessor, Computer Networks 31, April 1999, pp. 831–860.
[12] 平野 大輔、史 又華、戸川 望、柳澤 政生、”クロックグリッチに基づく故障解析に耐性を持つ AES 暗号回路”, 情報処理学会、システムと LSI 設計技術研究会, Vol. 2015-SLDM-171, No. 10, pp. 1-5, 2015.
[13] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten, “Lest We Remember: Cold Boot Attacks on Encryption Keys,” 17th USENIX Symposium, pp.45-60, 2008.
[14] 橋本幹生、春木洋美、川端健、”オープンソース OS と共存可能なセキュリティプロセッサ技術”、東芝レビュー、Vol. 60, No. 6, pp.44-47, 2005.
[15] 堀川健一、目崎元太、渡辺章代、加櫓武、松本達治、「自動車ソフトウェアの標準仕様”AUTOSAR”の評価」、SEI テクニカルレビュー、No. 175, pp.92-97、2009.