# FFRI Research Report

**2017 Vol.1**


株式会社 FFRI

# エグゼクティブサマリ

今回のリサーチレポートでは、FFRI が実施している「システムに対する脅威分析の調査・研究」とセキュリティカンファレンス Black Hat Asia, Black Hat USA 2017 で発表された最新セキュリティ研究の一部を紹介する。

脅威分析とは、システムやデバイスの仕様・設計に基づき、セキュリティインシデントや故障の原因となる脅威の有無を分析し、必要な対策を明らかにする作業である。要件定義や設計段階で脅威分析することで、開発工程の後期よりも効率的にセキュリティ対策が実施できる。また、セキュリティインシデントが発生した際、想定と対策を説明する根拠の 1 つにもなる。

しかしながら、脅威分析を実施するには、開発対象に関する知見とセキュリティに関する知見の両方が必要である。また、専門家でも多くの時間を要する。そのため、一般的な開発プロセスとして、まだ普及しているとは言えないのが現状である。

IoT 技術と既存技術を組み合わせた様々なシステムやサービスが開発されているが、それらにはサイバーセキュリティに対する懸念がある。安全なシステムを開発するために脅威分析を行うべきだが、規模や複雑さが増しているシステムの脅威を従来の手法で分析することが難しくなっている。

本レポートでは、システムに対する脅威分析を効率化するために FFRI とパナソニック製品セキュリティセンターが共同で検討した結果の一部を紹介する。

また、カンファレンスサーベイとして、2017 年 3 月下旬にシンガポールにて開催された Black Hat Asia 2017、及び 2017 年 7 月下旬にラスベガスにて開催された Black Hat USA 2017 における研究発表の中から、今後注目すべきセキュリティ技術に関する発表の概要をいくつか紹介する。

# 1. 基礎研究レポート システムに対する効率的な脅威分析手法の研究

## 1.1. 背景と目的

近年の ICT システムは様々な要素技術やデバイスなどを組み合わせ、利便性の高いサービスを提供している。これらのシステムの規模や複雑さは増しており、その全容が掴みづらくなっている。IoT 技術の普及により、この傾向は今後も継続すると考えられる。

一方、要素技術を組み合わせる過程、もしくは要素技術そのものに潜在する脅威により、システムは多くのサイバー攻撃や障害によるリスクを抱えている。安心・安全なシステムを実現するためには、脅威を分析する事で、リスクを軽減する必要がある。

我々は今回、脅威分析に関する先行研究の調査や、分析現場の現状把握を行った。その結果、既存の手法を用いた脅威分析には、セキュリティ専門家でも多くの時間を要する事や、分析者の経験や知識により、分析結果の深度に差が生じるという課題が存在する事が明らかになった。

これらの課題の解決を目指して、我々とパナソニック製品セキュリティセンターは、システムに対する効率的な脅威分析の手法に関する共同研究を行っている。本研究は脅威分析の網羅性や精度を追求する事が目的ではなく、より多くのシステム開発現場で脅威分析を実施可能にする事を目的とした研究である。

## 1.2. 本研究における脅威分析

一般に脅威とは、情報セキュリティマネジメントの規範を表した"JIS Q 27002：2006"において、「システム又は組織に損害を与える可能性があるインシデントの潜在的な原因」と定義されている[1]。

ここで本研究における脅威分析とは、分析対象とするシステムの設計や仕様に基づき、システム内に脅威が潜在していないか分析する事を指し、セキュリティテストとは異なるものである。

システム設計段階で脅威分析を実施する事で、脅威を早期に発見し、システム完成後に対応するよりもコストを抑えられる利点がある。その他にも、稼働済みのシステムに存在する脅威を明らかにするために行われる場合もある。脅威分析の結果は、システムにインシデントが発生した際、対策状況を説明する根拠の 1 つになる。

## 1.3. 先行研究

脅威分析に関する先行研究として、2 つの脅威分析手法を挙げる。

1 つ目は、Microsoft でプログラムマネージャーを務めていた Adam Shostack 氏が著書「Threat Modeling: Designing for Security」において提案している STRIDE による脅威分析手法である[2]。

STRIDE とは、情報システムにおける典型的な脅威である Spoofing （なりすまし）、Tampering （改ざん）、Repudiation（否認）、Information Disclosure（情報漏洩）、Denial of Service（サービス妨害）、Elevation of Privilege（権限昇格）の頭文字を取ったアクロニムである。

この手法では、システムの各要素に対し STRIDE の脅威を洗い出す事により脅威を発見する。ただし提案者の Adam 氏によると、STRIDE だけでは発見できない脅威も存在するとしている。

2 つ目は、ソニーデジタルネットワークアプリケーションズ株式会社の松並勝氏が提案している、設計図や仕様書からシステムのセキュリティ要件を明確化する事で、要件を満たしているか分析する手法である[3]。

この手法では、設計図や仕様書の説明を分析する事で「人」と「資産」を洗い出し、それぞれの動作(読み書き実行)と組み合わせる。これをもとに「誰が、何に、何を、できない（もしくはできる）」と言う観点からセキュリティ要件を決定し、脅威を発見する。

## 1.4. システム内の資産に着目した手法の検討

我々はまず、システムが保持する資産に注目して脅威分析を効率化する方法を検討した。

ここで資産とは、損失した場合に資産の所有者（当事者）が金銭的・心身的な負担を負うか、製造責任者（企業）が社会的な信用を失う有形及び無形のものを指す。

現在、様々な形態や規模のシステムが機能・情報の資産を保持している。それらの保持する資産が利用できなくなる場合や、資産を悪用される事から被害が発生する。

そのため、システムに存在する資産が少ない場合や、あるいは存在しても利用・悪用されて大きな被害に繋がらない場合、それらの部分に対する脅威分析は効果が低いと言える。

従って、脅威を分析する際、資産を基準として分析の範囲を限定する事で、効率化を図る事ができると考えられる。

そこで、我々は Data Flow Diagram(以下、DFD)を利用してシステムが保持する資産を評価し、その結果に基づいて分析の範囲を決定する手法を考案した。 図 1 に、模式図を示す。

本手法では加えて、脅威分析対象のシステムにおける資産の評価方法や、それに基づく分析の流れをフレームワーク化する事により、分析者のセキュリティ知識や経験への依存度を下げる事を目指した。

### 1.4.1. DFD の作成

本手法ではまず、DFD を作成する事でシステム全体の構造を明らかにし、各要素間でやり取りされるデータを列挙する。これにより、システム全体と構成要素が保持・関与する資産を洗い出す。本手法は資産に着目して脅威分析を実施するため、分析者は洗い出した資産の一覧表を作成する必要がある。

今回、DFD の作成と脅威の洗い出しを効率的に実施するために、Microsoft Threat Modeling Tool 2016(以下、TMT)を利用した[4]。TMT は DFD の作成・脅威の導出・脅威レポートの作成を補助するツールである。

このツールでは、脅威の発生条件を記述した脅威テンプレートを用いて、自動で脅威を洗い出す事ができる。作成した脅威テンプレートはカスタマイズ可能で、他の DFD でも再利用できる。

### 1.4.2. 各資産の重要度の評価

次に、洗い出した全資産の重要度を評価する。資産の評価は、各資産の持つ機密性(Confidentiality)、完全性(Integrity)、可用性(Availability)の 3 つの要素（以下、CIA）に着目し、それらが失われた際に生じる影響をもとに 8 段階で評価する。表 1-1 に、評価基準の詳細を示す。分析者は評価後、各資産の重要度を一覧表に記述する。

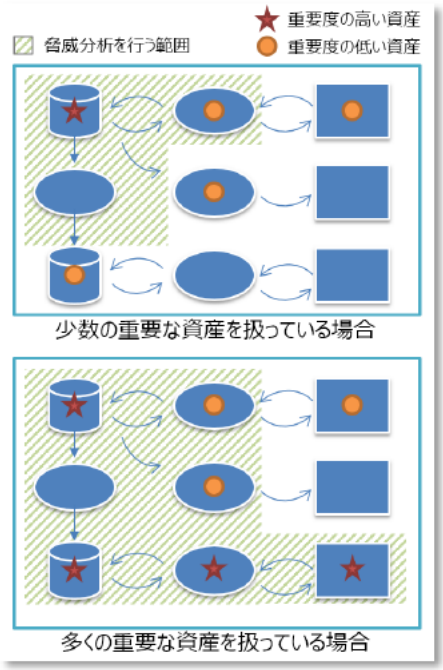


**図 1-1 資産の重要度に応じた分析範囲の設定**

### 1.4.3. システム全体の資産レベルの評価

次に、前工程で評価した各資産の重要度に基づき、システム全体が保有する資産のレベルを、低・中・高の 3 段階で評価する。分析者はこの結果から、システム全体に詳細な脅威分析を実施すべきか、簡易な脅威分析に留めるかを判断する。レベルの判断基準は次の通りである。

- レベル低システムの場合
  重要な資産をあまり扱っていないため、特に重要度の高い資産を扱う機能のみに脅威分析を実施する。
- レベル中システムの場合
  重要な資産をある程度扱っているため、重要度の高い資産を扱う機能全般に脅威分析を実施する。
- レベル高システムの場合
  重要な資産を多く扱っているため、著しく重要度の低い資産を扱う機能を除き、資産を扱う全ての機能に対して脅威分析を実施する。

システム全体の資産レベルの評価には、前工程で導出した各資産の重要度を CIA 別に合計し、各要素の平均値を利用する。これは、資産が多い巨大なシステムや、重要度の異なる様々な資産を持つシステムに対しても、適切に資産レベルを評価するためである。

ここで CIA 別平均値において、社会インフラやサービスの稼働に重大な影響を与える可能性のある閾値として、機密性は 4.0 以上、完全性は 3.5 以上、可用性は 4.0 以上と定義する。

システム全体の資産レベルの CIA 別平均値が、1 つでも閾値を超える場合、レベル高システムと評価する。

対して、システム全体の資産レベルの CIA 別平均値が、全て 2.0 に満たない場合はレベル低システムと評価する。

それ以外のシステムは、レベル中システムと評価する。

**表 1-1　CIA 別の資産評価基準一覧表**

| 資産評価 | 機密性の基準 |
|---|---|
| 5 | ・利用者の機微な個人情報、あるいは顧客から提示された機密情報<br>・大量の個人情報等、機密性が失われた場合に利用者の資産や社会的インフラに重大な影響が発生するもの |
| 4.5 | ・資産評価4の基準に該当する資産の内、単体機器に含まれているか、サーバーが保持するセキュリティ・金銭に関わるか、サーバーや機器を横断する顧客や著名に関わるもの |
| 4 | ・利用者の機微な個人情報、あるいは顧客から提示された機密情報<br>・機密性が失われた場合、利用者の資産もしくは社会的インフラの一部に影響が発生するもの<br>・機密性を担保する機能で使用されているもの |
| 3.5 | ・資産評価3の基準に該当する資産の内、単体機器に含まれているか、サーバーが保持するセキュリティ・金銭に関わるか、サーバーや機器を横断する顧客や著名に関わるもの |
| 3 | ・利用者の個人情報、あるいは顧客から提示された機密情報 |
| 2.5 | ・資産評価2の基準に該当する資産の内、単体機器に含まれているか、サーバーが保持するセキュリティ・金銭に関わるか、サーバーや機器を横断する顧客や著名に関わるもの |
| 2 | ・システム提供者（メーカ・販売・保守）と利用者のみが知りうる情報等 |
| 1 | ・公開情報 |

| 資産評価 | 完全性の基準 |
|---|---|
| 5 | ・常に完全な状態であることが必要<br>・完全性が失われた場合、利用者の生命、資産、もしくは社会的インフラに重大な影響が発生する |
| 4.5 | ・5の資産評価基準のいずれかを満たす |
| 4 | ・常に完全な状態であることが必要<br>・完全性が失われた場合、利用者の資産もしくは社会的インフラの一部に影響が発生するかサービスやシステムの継続的稼動に重大な影響が発生する |
| 3.5 | ・4の資産評価基準のいずれかを満たす |
| 3 | ・常に完全な状態であることが必要<br>・完全性が失われた場合、サービスやシステムの一部に影響が発生するか、機器の主要機能に影響が発生する |
| 2.5 | ・3の資産評価基準のいずれかを満たす |
| 2 | ・完全な状態に復元できることが必要ではない |
| 1 | ・復元できない場合でも支障がない |

| 資産評価 | 可用性の基準 |
|---|---|
| 5 | ・常にアクセスできることが必要（1分以内）<br>・アクセスできない場合、利用者の生命や資産、もしくは社会的インフラに重大な影響が発生する |
| 4.5 | ・5の資産評価基準のいずれかを満たす |
| 4 | ・常にアクセスできることが必要（1分以内）<br>・アクセスできない場合、利用者の資産もしくは社会的インフラの一部に影響が発生するか、サービスやシステムの継続的稼動に重大な影響が発生する |
| 3.5 | ・4の資産評価基準のいずれかを満たす |
| 3 | ・常にアクセスできることが必要（1分以内）<br>・アクセスできない場合、サービスやシステムの一部に影響が発生するか、機器の主要機能に影響が発生する |
| 2.5 | ・3の資産評価基準のいずれかを満たす |
| 2 | ・24時間以内にアクセスできることが必要<br>・アクセスできない場合、サービスやシステム、製品が提供する機能に影響が発生する |
| 1 | ・24時間以内にアクセスできなくても良い |

### 1.4.4. システムレベルに基づく分析範囲の決定

最後に、評価したシステムのレベルを基準として、システム内における分析範囲を決定し、当該機能に対して脅威分析を実施する。

レベル低システムでは、CIA の要素の内いずれかが資産評価一覧表の資産価値が 4.0 以上、「損失した場合、社会的インフラに影響を与える資産」を保有する機能を分析対象とする。

これに加えて、資産が損なわれた際、当事者が負う精神的苦痛と経済的損失の大きさも加味する。NPO 日本ネットワークセキュリティ協会セキュリティ被害調査ワーキンググループの資料[5]において、精神的苦痛レベル、又は経済的損失レベルの 3 にあたる資産を保有する機能も分析対象とする。

レベル中システムでは、レベル低システムで分析するものに加え、資産評価一覧表の完全性・可用性のどちらか資産価値が 3.0 以上、「損失した場合、システムの一部に影響が生じる資産」を保有する機能を分析対象とする。

レベル高システムでは、CIA 要素のいずれかが 2.0 以上である、「著しく価値の低い資産を除く全ての資産」に関係する機能を分析対象とする。

表 1-2 に、それぞれのシステムにおけるレベル別分析対象機能の条件を示す。

分析者は上記の基準を用いて、システムのレベルに合わせて分析対象の機能を決定する。これより脅威分析対象外の機能を決定後、TMT に設定を反映して脅威分析を実施する。

### 1.4.5. 仮想システムへの適用

実際に提案手法を仮想の POS システムに適用し、脅威分析を実施した。

システムは POS レジ端末、POS レジ端末の情報を集約する POS サーバー、POS サーバーに接続された売上確認用の端末等から構成される。売上確認用の端末はインターネットに接続しており、クレジット決済サーバーや POS サーバー管理端末との通信に利用する。図 1-2 に、本手法で作成した DFD を示す。

**表 1-2 システムレベル別の資産を分析対象とする条件**

| システムレベル | 資産を分析対象とする条件 |
|---|---|
| 低 | ・経済的損失が大きい資産<br>・漏洩した場合、当事者が精神的に大きな苦痛を受ける情報<br>・CIA別資産価値基準の内、下記のいずれかが基準値を超えている資産<br>- 機密性 … 4.0 / 可用性… 4.0 / 完全性… 4.0 |
| 中 | ・CIA別資産価値基準の内、下記のいずれかが基準値を超えている資産<br>- 可用性… 3.0 / 完全性… 3.0 |
| 高 | ・CIA別資産価値基準のいずれかが2.0を超えている資産 |

### 1.4.6. 課題と対策

提案手法を用いて脅威分析を実施した結果、従来手法を用いて脅威分析を実施した場合と比較し、分析対象範囲を大きく絞り込む事ができた。また、提案手法と従来手法の分析結果に大きな差は生じなかった。これらより、工数の減少により効率化を達成した。

実際に提案手法を用いて脅威分析を行った事で、以下の 3 つの課題点が明らかになった。

1 つ目の課題は、仕様書などの文章から資産を導出する際、列挙する資産の粒度や網羅性が、分析者や仕様書作成者の技量・経験に依存する点である。導出した資産一覧表における網羅性が不安定になる場合や、クレジットカードの PIN などの、一時的に利用するがシステム内に保有しない資産の記述漏れが生じやすい。

これらの課題については、明確な資産の定義や仕様書のガイドラインを策定し、一時的な資産は別途一覧表を作成する事により個別の対応が可能である。しかし、より抜本的な解決策が必要であり、現在検討中である。

2 つ目の課題は、資産重要度の評価において、特定の資産が別々の箇所で何度も導出される点である。提案手法では資産評価表の全ての資産に対して重要度を記入する必要があるため、同じ作業の繰り返しが生じた。これは、分析中に頻出する資産の重要度、及び脅威とその対策をテンプレート化する事で、負荷の軽減が可能である。

3 つ目の課題は、TMT を用いた脅威分析を行う際、DFD の各要素の記述や設定の誤りにより、システム設計上本来存在しない脅威を誤検出する点である。TMT によって導出された脅威は必ず正否判定が必要であるため、本来不要な作業がシステムの規模に比例して増大し、脅威分析の効率が低下する。DFD 作成時のヒューマンエラーが原因であり、解決策を現在検討中である。

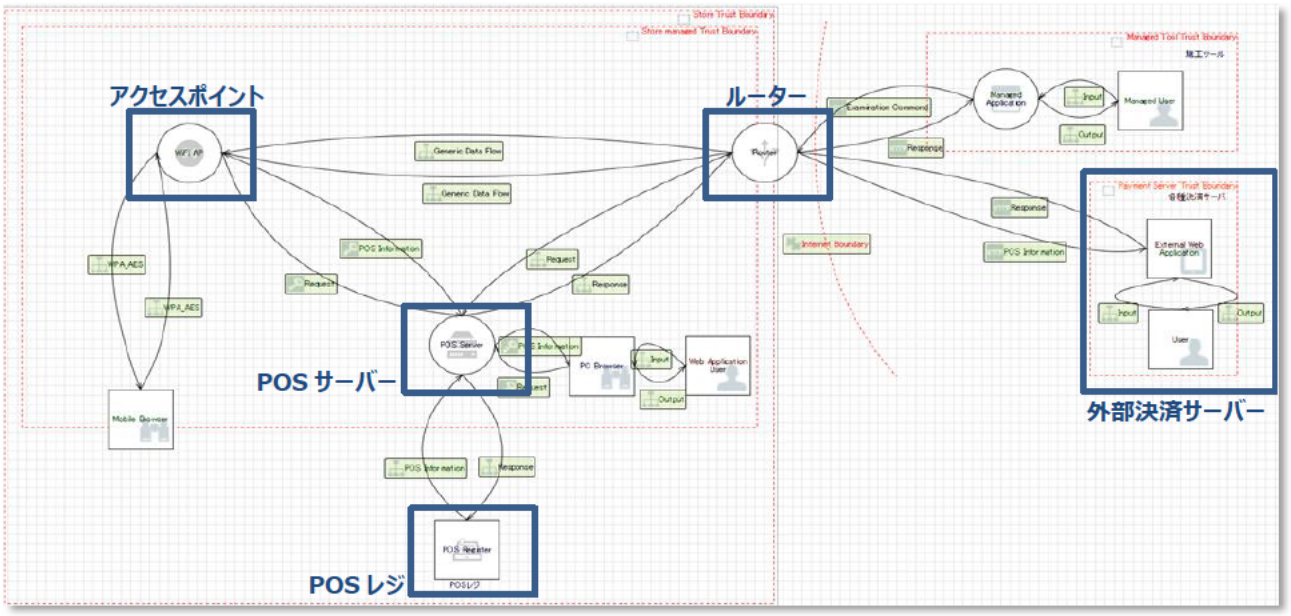


**図 1-2 システム内の資産に着目して作成した DFD**

### 1.5. 回避すべき事象に着目した手法の検討

別のアプローチとして、システムに生じうるクリティカルな被害に注目して脅威分析を効率化する手法を検討した。ここでクリティカルな被害とは、システムを利用したサービスに致命的なダメージを与えるものを指す。具体的には、重要な機器の乗っ取り、不正アクセスによる情報の窃取などが挙げられる。

本手法では、システムの構造を明らかにする事でクリティカルな被害が生じうる箇所を洗い出す。洗い出した脆弱な箇所を集中的に分析する事で、脅威分析の効率化を図る。

#### 1.5.1. システム開発者へのヒアリング

本手法ではまず、分析対象とするシステムの開発者に対しヒアリングを実施し、システムに生じうるクリティカルな被害を想定する。

ここで、想定した被害から導き出される、守るべき資産や機能の総称をターゲットと定義する。

本手法においてターゲットは複数設定してもよく、必要に応じて CIA や STRIDE 等の観点を含めてもよい。

#### 1.5.2. DFD の作成

次に、仕様書や設計書等のシステム設計書類をもとに、システム全体の DFD を作成する。想定したクリティカルな被害に関連するか否かに係わらず、システム内の全ての要素を記述する。

図 1-3 に、本手法の手順により作成した DFD を示す。仮想システムの構成は、1.4 章における POS システムと同様である。

本手法における DFD の作成手順は、次の通りである。

① 大まかな DFD の作成

記述するシステム内の各要素の粒度はモジュール単位とし、デバイスや建物などの物理的境界とドメインや有線無線などのネットワーク境界は、全て信頼境界として記述する。

② ユースケースごとに関連要素を絞った DFD の作成

考えられるユースケースごとに、関連要素のみを記述した DFD を作成する。記述するデータの粒度は、それぞれの性質を考慮して記載する。例えば、認証情報とユーザーデータが同時に流れる場合、用途が異なるデータであるため別々の要素として記載する必要がある。

③ ユースケースごとに作成した DFD の結合

前手順で作成した DFD を、1 つの DFD に統合する。ここで DFD を結合する際、簡略化の前準備として右端あるいは左端にターゲットを配置する。

複数のターゲットが存在する場合は、ターゲットごとに DFD を統合、作成する。各要素の粒度は予め決定しているため、同じトポロジーを持つ DFD を複数作成する必要がある。

④ デバイス・ネットワーク境界・情報をグループ化

前手順で作成した DFD は、各要素に対して複数の機能が列挙されている。機能や情報をグループ化する事で、簡略化や抽象化の準備を行う。

⑤ 統合した DFD の簡略化と抽象化

同一デバイス内の機能を統合する。想定している仮想システムであれば、POS サーバー内の通信中継機能や、売上管理機能などの要素を統合し、単一の POS サーバーとする。

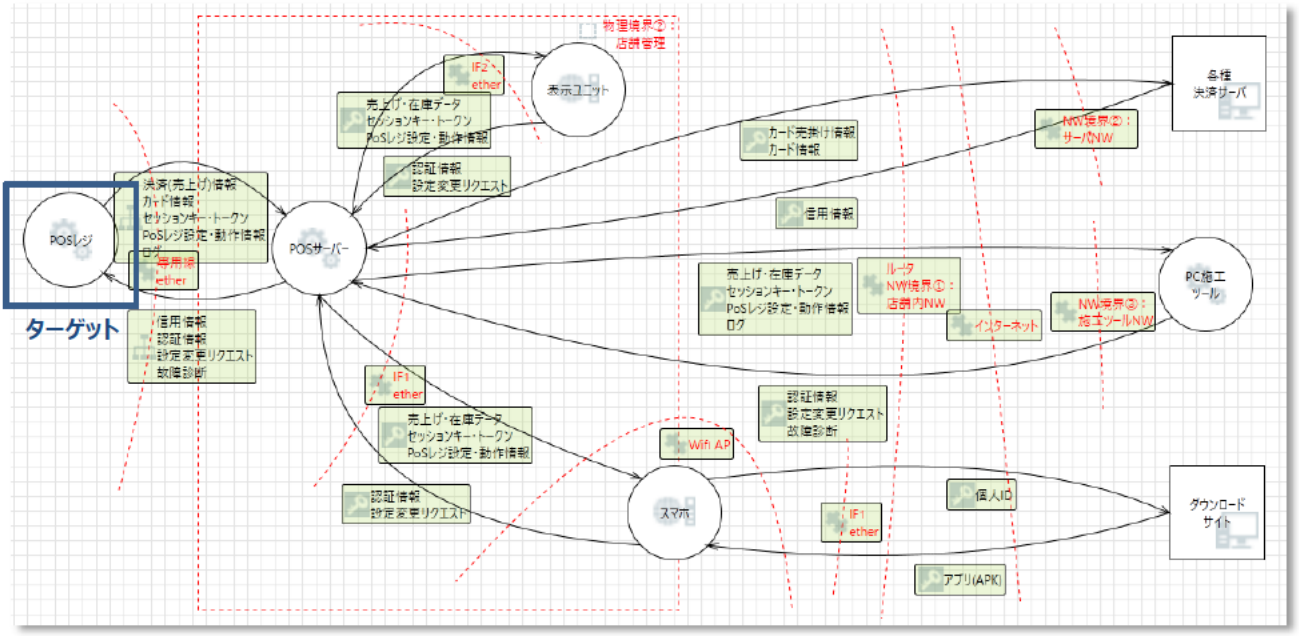


**図 1-3 回避すべき事象に着目して作成した DFD**

### 1.5.3. Attack Tree の作成

最後に、統合した DFD をもとに Attack Tree を作成する事で、ターゲットへの経路をたどる際に必要な条件を洗い出す。

まず、ターゲットを Attack Tree 上の頂点に配置し、DFD の各要素と経路、及び物理アクセスをノードとして記述する。経路や要素をたどる際に必要となる前提条件が存在する場合、その旨も記述する。

Attack Tree 中において、経路をたどる条件は基本的に OR である。これは、いずれかの経路でターゲットに到達可能であれば、クリティカルな被害が生じうるためである。

本手法で作成した Attack Tree により、クリティカルな被害が生じる原因となる脅威が可視化され、対策を検討しやすくなる。

### 1.5.4. 課題

実際に提案手法を用いて脅威分析を行った結果、分析者への属人性が高い点が明らかになった。DFD の作成・結合の工程において、分析者にある程度の経験や技能が必要となる。

また、DFD 作成の際、巨大なシステムなどにおいては非常に多くのノード・経路が洗い出される事が想定される。これにより Attack Tree が複雑となり、脅威分析の効率化が達成できない可能性がある。これに関しては抜本的な解決策が必要であり、現在検討中である。

## 1.6. まとめ

本共同研究では、システム内の資産に着目して脅威分析の範囲を決める手法と、システムにおいて回避すべき事象に着目し、脅威分析の範囲を決める手法の 2 つを検討した。表 1-3 に、それぞれの手法の特徴を示す。

どちらの手法も DFD の作成を起点としているため、まずは DFD の作成を効率化する事が重要である。また、どちらの手法も属人性に関する課題が存在する。

今後の研究において、手法の簡略化や脅威分析の専門家が用いる手法をナレッジとして明文化する事で、解決できると考える。

脅威分析の研究で感じる事は、脅威分析結果の公開事例が少ないという点である。この原因は、分析結果を公開する事により対策箇所が攻撃者に推測され、セキュリティリスクが増加する事を避けるための判断だと考えられる。

そうした中、脅威分析の情報共有や意見交換を通して理解促進・発展普及を目指す脅威分析研究会/SIGSTA[6]が 2016 年から発足されている。文献の解説資料や、様々な業種における脅威情報や分析テクニックが共有されており、本研究においても非常に参考になった。

**表 1-3 提案した手法の特徴**

| 手法 | システム内の資産に注目した手法 | 回避すべき事象に着目した手法 |
|---|---|---|
| 目的 | 重要な資産を持つシステムに対する集中的な脅威分析 | クリティカルな被害の防止 |
| 準備 | 開発資料からDFDを作成し、分析対象の資産を洗い出す。 | 開発者にヒアリングを行い、クリティカルな被害の想定に基づきDFDを作成する。 |
| 脅威分析 | TMTの脅威分析機能を用いる。 | Attack Treeにより脅威の可視化を行う。 |
| 特徴 | システムの持つ資産の重要度に基づき、脅威分析を行う範囲を決定する。<br>最も負荷が大きく、属人性の高いタスクは、「システム内の資産の導出と分析」である。 | クリティカルな被害の想定は、開発者に決定権がある。<br>最も負荷が大きく、属人性の高いタスクは、「DFDの作成」である。 |
| 課題 | 過不足なくシステム内の全資産を導出することが難しい。<br>ヒューマンエラーにより、誤検出が生じる。 | DFDの作成・結合の際、分析者に技能・経験が要求される。<br>システムの規模に比例し、作成するDFDやAttack Treeの量、規模が増大する可能性がある。 |

# 2. カンファレンスサーベイ Black Hat Asia 2017

本章は、2017 年 3 月 28 日から 31 日にかけてシンガポールで開催された、世界的なセキュリティカンファレンスである Black Hat Asia 2017[7]のサーベイレポートである。30 件以上の発表や、高度なセキュリティ技術を学ぶワークショップが開催された。

## 2.1. 攻撃解説

### データ指向攻撃の手法とその対策

#### 2.1.1. 発表の概要

「The Power of Data-Oriented Attacks: Bypassing Memory Mitigation Using Data-Only Exploitation Technique Part I」[8]では、Data-Oriented Attacks と呼ばれる攻撃方法を利用する事で、Microsoft Edge に対して Silverlight を不正に追加する実証を行っている。

2016 年 12 月に Microsoft からリリースされた Windows Insider Preview build 14986 から Control Flow Guard（Guard CF）が強化され、それまで Windows で外部の任意コードを呼び出す攻撃に有効であった、多くのコールターゲットが無効化された。これにより、Windows において任意コードの実行による攻撃の難度が格段に上昇した。

そこで別の手法として、Data-Oriented Attacks と呼ばれるプログラムの制御フローではなくデータを操作する事により、動作を変更する攻撃が注目を集めている。プログラムの制御フローを操作しないため、Guard CF などの CFI（Control Flow Integrity）が適用されている環境においても攻撃が可能となる。

現在、多くのブラウザにおいて Silverlight などのメディア実行技術は脆弱性が生じる温床となっている。しかし HTML5 拡張によるセキュアなメディアソリューションが登場した事により、Microsoft は新ブラウザの Microsoft Edge（以下、Edge）以降では、Silverlight への対応を行わない事を表明している。

本発表では、データへの攻撃のみ（Data-only-Attack）で、Edge に対して ActiveX コントロールの一種である Silverlight プラグインを強制的に読み込み、攻撃面を増やす実証を行う。

Edge に Silverlight プラグインをロードする際、2 つの技術的関門が存在する。

1 つめは ActiveX オブジェクトがインスタンス化される際のチェック機能、2 つめは LoadLibraryExW 関数を用いてライブラリを呼び出す際の引数である。以下に、それぞれの詳細を紹介する。

- **ActiveX オブジェクトがインスタンス化される際のチェック**

Edge では、ActiveX のオブジェクトをインスタンス化する際に、オブジェクトの安全性を確認するために 2 つのセキュリティチェックを行っている。これらのチェックは最終的に、urlmon.dll により提供される urlmon!g_pFeatureCache というフィーチャーキャッシュ内の 14 番目のエントリを参照し、拡張機能が利用可能であるか確認を行っている。

発表によると、このフィーチャーキャッシュの存在するメモリ領域は書き込みが禁止されておらず、任意の値に書き換えが可能であった。実際に上記のエントリを書き換え、チェックを回避できる事が実証されている。

- **LoadLibraryExW 関数の引数**

Edge において Silverlight プラグインモジュール(npctrl.dll)を読み込む際に LoadLibraryExW 関数を用いている。この関数を呼び出す際、プラグインモジュールの絶対パスと、モジュールを読み込む際の動作を指定する引数である dwFlags として、"LOAD_WITH_ALTERED_SEARCH_PATH"を設定する。これらの条件で関数を呼び出す事で、モジュールが読み込まれる。ここで、"LOAD_WITH_ALTERED_SEARCH_PATH" は combase.dll で定義されるグローバル変数（combase!g_LoadLibraryAlteredSearchPathFlag）である。即ち、LoadLibraryExW 関数を上記の条件で実行する場合、それに伴い combase.dll が読み込まれるという事である。

発表者が着目したのは、上記グローバル変数が存在するメモリ領域に対するアクセス権限である。combase.dll のデータセクションに定義されているこの変数は、読み込まれた後のメモリ領域が書き込み禁止されておらず、任意の値に書き換えが可能であった。

これにより LoadLibraryExW 関数を経由した、Silverlight プラグインモジュールの強制的な読み込みが可能である。

### 2.1.2. 攻撃への対策

今回の発表では攻撃手法とその実証と同時に、対策手法も提案されている。

Silverlight プラグインの読み込み自体は、ACG（Arbitrary Code Guard）を有効にする事でブロックが可能である。しかし、これは LoadLibraryExW 関数の引数に関する問題には対処できるが、ActiveX オブジェクトがインスタンス化される際の制限に関する問題に対処できない。

従って Windows アプリケーションの開発者には、重要なデータ（変数）は書き込みを禁止するか、禁止が難しい場合は強固なアルゴリズムによって暗号化し、秘密鍵をカーネル空間において保護する事が望まれるとしている。

### 2.1.3. 考察

データ指向攻撃は、CFI（Control Flow Integrity）技術の向上に伴って増加傾向である。既に Windows や Linux の環境に対して、データ指向攻撃を行うエクスプロイトを自動生成する試みなども発表されている[9]。

ARM 系アーキテクチャで動作する Linux 系 OS においては、権限の管理は SELinux を導入し、カーネル空間の代わりに TrustZone などを利用することで、本攻撃手法で用いられているメモリ空間への不正アクセスを防止できると考えられる。

また、発表者は HITBSecConf 2017 AMSTERDAM において「The Power of Data-Oriented Attacks: Bypassing Memory Mitigation Using Data-Only Exploitation Technique Part II」[10]と題し、Windows で DLL の遅延読み込みを行うためのライブラリに対して、データ指向攻撃を仕掛ける手法も発表している。

## 2.2. マルウェア解析技術

### Hollow Process Injection の発見と回避

### 2.2.1. 発表の概要

「What Malware Authors Don't Want You to Know - Evasive Hollow Process Injection」[11]は、Hollow Process Injection を VAD と PEB の矛盾をもとに検知する技術についての発表である。

Stuxnet などを始めとする Hollow Process Injection を使用したマルウェアを見分ける方法、及び確認すべき項目を解説し、それらの解析を行う為の補助ツールが紹介された。

Hollow Process Injection は Process Hollowing とも呼ばれ、正当なプロセスのセクションの中身を空洞（Hollow）状態にし、そこに攻撃コードを書き込む攻撃を指す。この手法を利用したマルウェアには、Stuxnet、Skeeyah などが挙げられる。

PEB（Process Environment Block）とは、プロセスが利用している DLL 等のリソース情報を保持するデータ構造体であり、32 ビット版 Windows においては fs セグメントの 0x30 を参照する事でアクセスが可能である。

VAD （Virtual Address Descriptor）とは、カーネルメモリに存在する、連続したプロセス仮想アドレス空間の情報を保持するツリー構造体である。

本来、これら 2 種類のメモリで保持されるプロセスのベースアドレスは同一である。ここで Hollow Process Injection では、プロセスのリソース情報を書き換える事で攻撃コードを実行させる。変更されるのは PEB のみであるため、VAD と比較した際に矛盾が生じる。本発表では、主にこの特性を利用し攻撃を検知する。

### 2.2.2. 実際のマルウェアの検知

今回の発表では、Stuxnet と呼ばれる Hollow Process Injection を用いたワームを検出する方法が紹介された。具体的な検出の手法は、次の通りである。

- **不審なプロセスの発見**

最初に、Stuxnet が擬態していると考えられる不審なプロセスを発見する。Windows のシステムプロセスである lsass.exe は、Vista 以前の Windows では winlogon.exe、Vista 以降は wininit.exe を親プロセスとして動作する。Stuxnet が動作している環境では、親プロセスが services.exe となっている 2 つの lsass.exe が存在する。通常、lsass.exe のプロセスは 1 つしか存在しないため、どちらかが不審なプロセスであると考えられる。

- **PEB と VAD の比較**

メモリフォレンジックツールである Volatility Framework を用いて、PEB と VAD の比較を行う。PEB と VAD の確認にはそれぞれプラグインが必要であり、PEB には dlllist プラグインを用い、VAD には ldrmodules プラグインを用いる。

Stuxnet が動作している場合、PEB と VAD を比較すると次に示す 2 つの矛盾点が確認できる。

① VAD のプロセスパスの項目に何も表示されない点

② 各プロセスのベースアドレスが、PEB と VAD で異なる点

PEB はプロセスメモリに存在しており、プロセスが書き換えられた際に、保持しているプロセスのベースアドレスも変更される。

対して、VAD はカーネルメモリでアドレス空間の割当を保持しているため、プロセスが書き換えられてもベースアドレスは変更されない。

これらより、Stuxnet が動作している場合、上記の矛盾が生じる。

さらに、Volatility Framework で提供されている malfind プラグインにより、プロセスの Vad Tag を確認する。

正当な実行ファイルの場合は Vad Protection の値が"PAGE_EXECUTE_WRITECOPY"であるが、Stuxnet が動作している場合は、"PAGE_EXECUTE_READWRITE"となる。これらの矛盾点から、発見した不審なプロセスは正当なプロセスでないと判断できる。

### 2.2.3. 検出ツール

発表者は、一連の処理をワンステップで確認可能で、さらにダンプ機能が実装された、Volatility Framework のプラグイン「HollowFind」を紹介した。

今回の発表では Stuxnet の他に、冒頭で紹介した台湾での標的型攻撃に使われたマルウェアや、Skeeyah、Kuluoz Kronos など、様々なマルウェアの解説を行っている。

### 2.2.4. 考察

Hollow Process Injection は多くのマルウェアで利用されており、本検出ツールを用いる事で分析がより容易になる。ただし、発表者が開発した Volatility Framework プラグインは、ダンプファイルに対して解析を行うツールである。そのため現時点にでは、マルウェアの検知ではなく、検出した検体の分析に利用できる。

## 2.3. Android セキュリティ

### プラグイン技術によるアプリの実行を検出する

### 2.3.1. 発表の概要

「ANTI-PLUGIN: DON'T LET YOUR APP PLAY AS AN ANDROID PLUGIN」[12]では Android プラグイン技術の検出を行う研究を発表された。

Android のプラグイン技術では、ホストアプリケーション（以下、ホスト）内に、ゲストとなるアプリケーション（以下、ゲスト）のパッケージファイル（APK ファイル）を保持する。ここでゲストとなる APK ファイルは、同一のものが複数存在してもよい。ホストはサンドボックスを用意し、その中でゲストを実行できる。

これらの性質から、プラグイン技術は主に Parallel Space などの、異なるサービスのアカウントを包括して運用するアプリケーションで利用されている。

ホストはネットワークにおけるプロキシのように、ゲストが利用している Android API 呼び出しのフックが可能である。

即ち、不正なホストが正規のゲストから呼び出された API 呼び出しをフックする事で、ユーザーのログイン情報等を盗聴する事などが可能である。2016 年 12 月には、パロアルトネットワークス社によるプラグイン技術を使ったマルウェア PluginPhantom のレポートが発表されている。

攻撃者がこの技術を使う利点は、正規のパッケージファイルに対するリパッケージを行わずにフックが可能であるにも係わらず、プラグイン技術自体は正当な機能であるため、静的な検知がされづらいという点である。

### 2.3.2. プラグイン技術の悪用からアプリを守る

発表者は、プラグイン技術の悪用を検知するための、Plugin-Killer というライブラリを考案した。

このライブラリを用いてアプリケーション内にチェックメソッドを追加する事で、プラグイン技術により作成されたサンドボックス内でゲストが動作しているか調査できる。アプリケーションの開発者は、その結果をもとに、プラグイン技術への対応策を実装できる。

Android では、パッケージごとに異なるユーザーのプロセスとして動作する。即ち、パッケージごとに個別の UID が付与される。

ネイティブ環境では、アプリがインストールされた段階で、システムにより UID が設定される。対して、サンドボックス内で動作する場合は、サンドボックス環境のホストにより UID が設定される。これにより、ネイティブ環境とプラグイン技術のサンドボックス環境において差異が生じる。

また、ホストと全てのゲストは、Android システムからは単一のパッケージとして認識される。

これらの点に注目する事で、ゲストの動作の有無を検出可能である。具体的な検出方法として、次の 4 点が挙げられる。

① アプリケーションのマニフェスト項目の確認

本来必要なシステム権限よりも多くの権限が付与されている場合や、アプリコンポーネント名が異なる場合。

② アプリケーションを実行しているユーザー（UID）の確認

ホストとゲストは UID を共有している事が多いため、実行中プロセスの一覧において UID の重複が確認できる場合がある。

③ アプリケーションコンポーネントの確認

サービスとスタブアクティビティについて、ホストは事前に一定数のスタブを予め用意している。これらのスタブは、ゲストが外部と通信する際に利用している。そのため、用意されたスタブの数を越えるサービスを開始した場合に、スタブの不足よる通信動作の不調が生じる。また、静的なブロードキャストレシーバーも、ネイティブ環境とサンドボックス環境で挙動が異なる。

④ 他のゲストアプリが生成したファイルの確認

ブラウザクッキーなどの情報を共有しているため、他のゲストアプリが生成したファイルを確認できる場合がある。

### 2.3.3. 考察

Android で動作するアプリケーションを開発する際には、今回紹介されたライブラリのような技術を用いて、不正なゲストの有無を調査する実装が望ましい。

ユーザーは複数のアカウントを管理できるアプリケーション等の利用に際して、アプリケーションの要求する権限や提供元を確認する事が重要である。

# 3. カンファレンスサーベイ Black Hat USA 2017

本章は、2017 年 7 月 22 日から 28 日にかけてラスベガスで開催された、世界的なセキュリティカンファレンスである Black Hat USA 2017[13]のサーベイレポートである。1997 年に初めて開催された Black Hat は今年で 20 回目を迎え、過去最高の 15,000 人以上の参加と 100 件以上の発表があった。IoT 機器のセキュリティやマルウェア（ランサムウェア）をテーマとした発表が多数見られた。

## 3.1. ランサムウェアに対抗するファイルシステム

### 3.1.1. 発表の概要

「ShieldFS: The Last Word in Ransomware Resilient File Systems」[14] は、大流行しているランサムウェアからデータを保護する事を主眼に置いて開発された、Windows 向けファイルシステムについての発表である。

ランサムウェアに感染すると、コンピュータに保存されたユーザーのデータが暗号化され、復元に対して身代金を要求される。暗号化されたデータは、基本的には身代金を払う事でしか復元できない。中にはデータを復元する手段がないにもかかわらず、ランサムウェアと同じく身代金を要求するマルウェアも確認されている。

発表者らにより開発された"ShieldFS"では、ランサムウェアの検知と停止、及び暗号化されたデータの復元を実現している。

以下に、本発表の詳細を紹介する。

### 3.1.2. ShieldFS の実装

- **ランサムウェアの検知機能**

ShieldFS は、"FS Activity Monitor"と言うWindows カーネルモジュールを実装している。これはファイルシステムへのアクティビティの監視と、それらのロギングを行うモジュールである。
本モジュールは Windows Minifilter Driver として実装され、ファイルシステムへの IRPs（I/O Request Packet）を監視している。

発表者らは、正常なアプリケーションとランサムウェアにおけるファイルシステムへのアクティビティ、すなわちアクセス要求の内容などにおける差異に着目した。この差異について、機械学習を用いて分類器を生成する事で、ランサムウェアの検知を行うシステムを実装している。

ShieldFS ではプロセスのファイルシステムに対するアクティビティを識別する際、アクティビティに対して複数の分類器を同時に適用する多層構造を取っている。図 3-1 に、概略図を示す。

ファイルシステムへのアクティビティを N%の範囲で監視する分類器、2N%の範囲で監視する分類器、3N%の範囲で監視する分類器、6N%の範囲で監視する分類器の 4 層で構成されている。それぞれの分類器でプロセスの動作を監視して不審な動作の有無を判定し、全ての分類器の多数決を取る事で、最終的にプロセスを停止するか決定している。

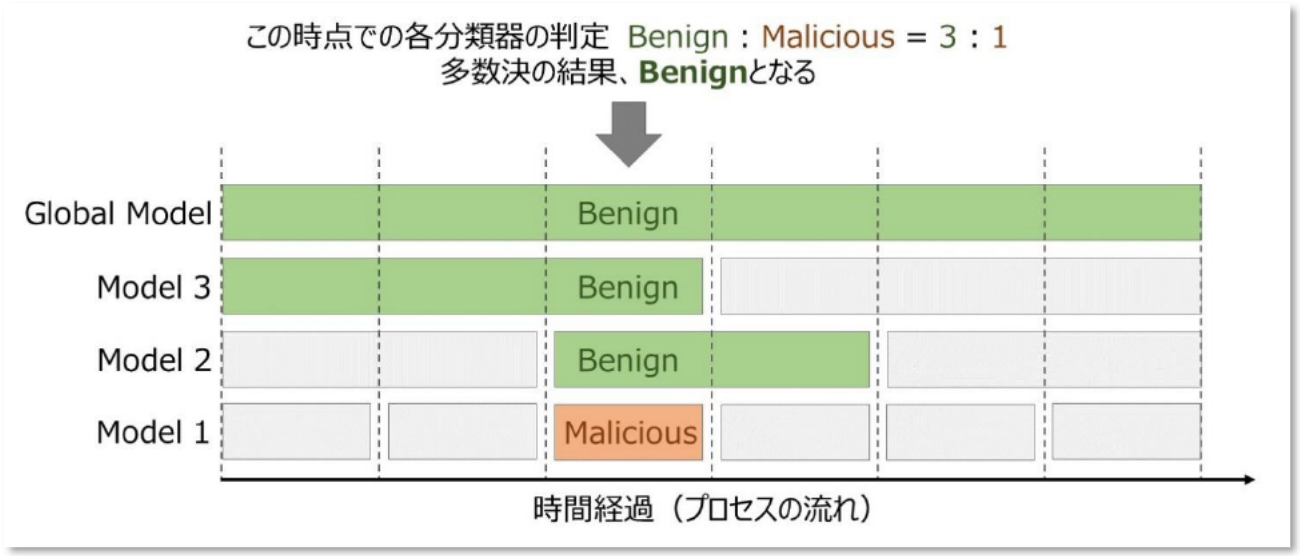


**図 3-1 多層構造を持ったマルウェア検出器**

- **暗号化されたデータの修復機能**

ShieldFS では、実際にデータが存在するファイルシステムとプロセスの間において、バッファのように動作する。

プロセスがファイルシステムに対して初回の書き込みを行った際、ダミーファイルを生成する。このダミーファイルに対してプロセスからの書き込み命令を適用しており、実際のファイルシステムには即座に変更を加えない。

前述の検知機能により、プロセスが不審でないと判定した場合、オリジナルのデータに対して変更を適用する。対して、プロセスが不審であると判定した場合は、プロセスを停止した後、オリジナルのデータを復元する。これにより、ユーザーのデータを保護している。

### 3.1.3. 検証結果

分類器生成の際、学習用データセットに含めなかったランサムウェア群を用いて、検知とデータの保護が有効であるか確認を行っている。

1483 個のランサムウェアをテストデータセットとして用いた結果、ランサムウェアの検知率は 96.9%、データの復元率は 100%であったとの事である。

同時に、正常なアプリケーション群を誤ってランサムウェアとして検出する確率（誤検出率）にも言及している。11 台のコンピュータで検証した結果、9 台のコンピュータで 0%、1 台で 0.27%、残る 1 台で 0.15%であったとの事である。

### 3.1.4. 考察

2017 年 5 月に大流行したランサムウェア"WannaCry"は、国内でもいくつかの感染事例が報告されており、対策の必要性は増してきている。今後はユーザーのデータ保護を含めた、セキュリティ対策技術が求められる。

本発表にあったShieldFSは、マルウェアの検知とデータ保護を同時に行う事ができる、非常に有用な技術である。

しかし同時に、事後対策としてファイルの復元を行うため、ShieldFS はある程度のリソースを要求すると考えられる。また、実装の仕様上、遅延書き込みを行う性質を持っている。そのため、想定外のシステム障害などが生じた場合、書き込み途中のデータが失われる可能性も考えられる。

今後はIoT機器などの小さなシステムを対象としたマルウェアも増加すると考えられ、データの復元等の事後対策を十分に行う事が物理的に不可能なケースも想定される。

従って今後は、事後対策だけでなく、現状よりもさらに軽量かつ高精度なマルウェア検知技術が必要になると考えられる。

## 3.2. Wi-Fi チップセットの脆弱性解説

### 3.2.1. 発表の概要

「Broadpwn: Remotely Compromising Android and iOS via a Bug in Broadcom's Wi-Fi Chipsets」[15]は、Android デバイス、iOS デバイスにも利用されている Broadcom 社の Wi-Fi チップセット"BCM43xx"ファミリにおける、遠隔攻撃に利用可能な脆弱性に関する発表である。

本発表で述べられている Wi-Fi チップセットは SoC であり、内部にプロセッサとメモリを持つ。デバイスの起動時にファームウェアを読み込み、ホットスポットとのやり取りなどを処理している。

本脆弱性の発見方法や、Wi-Fi チップセットへの脆弱性攻撃をメインプロセッサに連鎖させ、システム全体に影響を与える手法について述べられている。

### 3.2.2. 遠隔攻撃

遠隔攻撃はしばしば、「Web ブラウザを通じた能動的攻撃である」と考えられているが、被害者にリンクをクリックさせる等の受動的プロセスを必要とする時点で、完全な遠隔攻撃ではないと発表者らは確認している。発表者らにより定義される遠隔攻撃は、能動的攻撃を指しており、次の 3 つの要素を満たす必要がある。

- 被害者による操作を攻撃のトリガーとしない。
- 攻撃実行の要件として、攻撃対象のシステム内部に複雑な条件を必要としない。
- 攻撃実行後、システムは継続的に安定動作しなくてはならない。

### 3.2.3. Wi-Fi チップセットへの脆弱性攻撃

Android や iOS の場合、OS 側に ASLR（アドレス配置ランダム化）や DEP（データ実行防止機能）などが実装されている。

ASLR は、メモリ空間中の重要なデータ領域の位置をランダム化することで、攻撃を妨害する。DEP は、メモリ領域に対して実行可否のラベリングを行うことで、攻撃コードの不正な実行を防止している。

これらの働きにより、システムの内部状態は攻撃者が推測しづらく、かつ攻撃しづらいセキュアな状態となっている。従って、ソフトウェアに対する遠隔攻撃は非常に難易度が高い。

対して、ハードウェア側の Broadcom 社製 Wi-Fi チップセットでは、DEP も ASLR も実装されていない。加えて、メモリ空間の任意の領域に対して、読み書き実行が可能となっていた。
本攻撃手法の概要を、図 3-2 に示す。

これより発表者らは、Wi-Fi チップセットを標的として攻撃手法を検討した。先に述べた遠隔攻撃に必要な 3 つの要素に従い、Wi-Fi チップセットのファームウェアのリバースエンジニアリングを通じ、以下の調査検討を行っている。

- 攻撃面の検討

発表者らは、Wi-Fi チップセット内部の"Wireless Media Extensions[WME]"実装部におけるメモリ確保を行う関数部に、バッファオーバーフローを伴う脆弱性を発見した。

ここで WME とは、802.11 標準における QoS（Quality of Service）に関する拡張で、VoIP や動画などのリアルタイム性を必要とする通信パケットを優先的に送受信するための規格である。

- 攻撃実行時のシステム内部状態

攻撃可能なシステム内部状態の条件は、「先述の脆弱性に対して攻撃を行った場合、静的メモリ領域で確実にバッファオーバーフローが生じる」事が必要である。これは、動的メモリ領域でバッファオーバーフローが生じた場合、シェルコードを実行する際に使用するアドレスが毎回変化してしまい、攻撃が成功しづらくなるためである。

最終的な調査結果では、攻撃に利用可能なオーバーフロー領域はわずか 24 バイトであると判明し、それ以下のサイズでシェルコードを書き込む必要があった。発表にあったシェルコードを、コード 3-1 に示す。

- 攻撃後のシステム安定性

攻撃成功後は、メインプロセッサで動作する OS に攻撃を連鎖させるため、Wi-Fi チップセットを通じてマルウェアを侵入させる必要がある。特定のパケットを受信した際、先の攻撃で生成したエクスプロイトコードを実行する事で、パケット内に含まれるマルウェアのデータを抽出する事により、システム内にマルウェアを侵入させる事に成功している。

なお、Wi-Fi チップセットのファームウェアはいくつかのバージョンが存在するため、それぞれのバージョンに応じた攻撃が必要になる場合もあるとの事である。本攻撃手法から、システムカーネルへの攻撃を連鎖させることが可能である事が示唆されている。

具体的なシナリオとして、PCIe を介してメインメモリの直接書き換えが行える可能性が、Project Zero（Google）により言及されている。その他にも、Wi-Fi チップセットにおいてトラフィックを書き換えることにより、不正な URL へリダイレクトさせ、ブラウザの持つ脆弱性と組み合わせることで、メインカーネルへの攻撃が可能であると示されている。

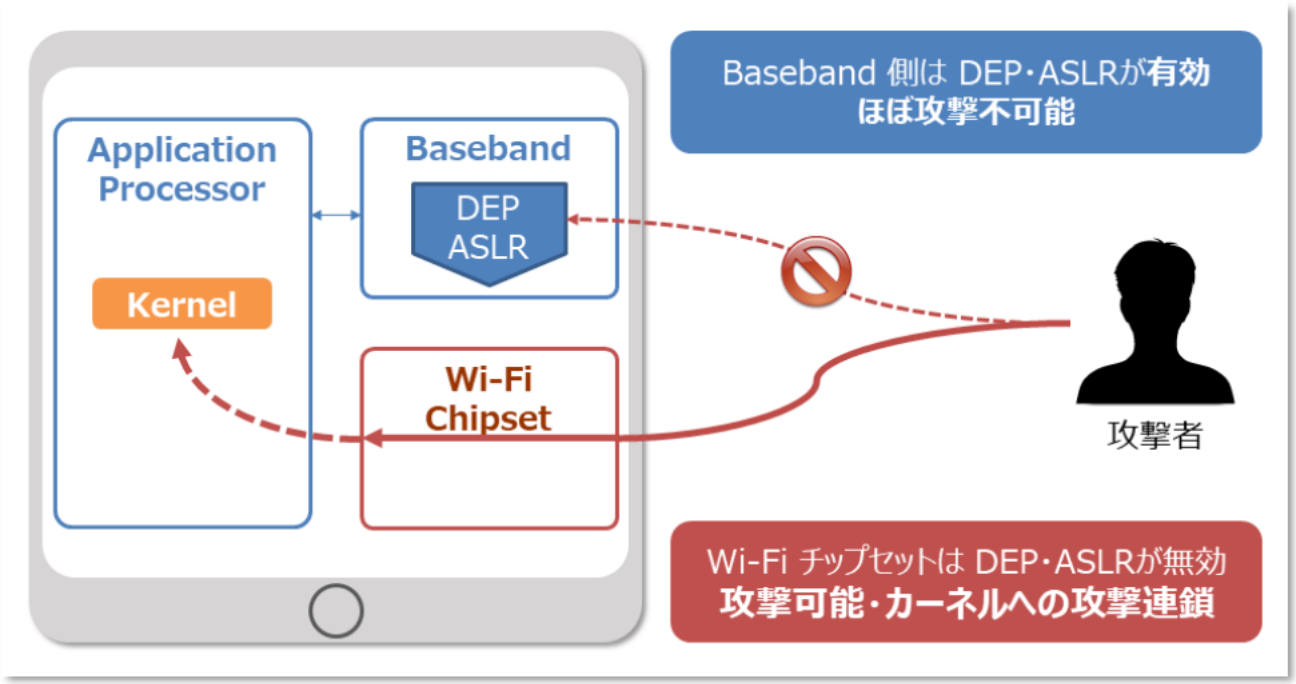


**図 3-2 Wi-Fi チップセットを経由したシステムカーネルへの攻撃**

```
__attribute__((naked)) void shellcode_start(void)
{
    asm(
        "push {r0-r3,lr}¥n"
        "bl egghunt¥n"
        "pop {r0-r3,pc}¥n"
    );
}

void egghunt(unsigned int cpsr)
{
    unsigned int egghunt_start = RING_BUFFER_START;
    unsigned int *p = (unsigned int *) egghunt_start;
    void (*f)(unsigned int);
loop:
    p++;
    if (*p != 0xc0deba5e)
        goto loop;
    f = (void (*)(unsigned int))(((unsigned char *) p) + 5);
    f(cpsr);
    return;
}
```

**コード 3-1 発表の実証に利用されたシェルコード**

### 3.2.4. 考察

最も普及している最先端のデバイスで用いられている SoC に存在する脆弱性に関する発表であり、非常に影響の大きい問題である。

本攻撃手法をそのまま流用するのみであっても、ブラウザへの攻撃を連鎖させる事で、Wi-Fi チップ上でユーザーのトラフィックを傍受・改ざんを行う MITB 攻撃などにも応用が可能である。また、Wi-Fi チップは PCIe を介して DRAM へ直接接続されているため、DMA（Direct Memory Access）を利用することで DEP を回避し、メインプロセッサで動作する OS に対する攻撃も可能であると考えられる。ASLR の回避については、本攻撃手法のみでは不可能である。

IoT の隆盛に伴い、別の小型デバイスやコネクテッドカー、チップセットや機器についても、同様の脆弱性が存在する可能性がある。これらについても Android や iOS 同様、堅牢なセキュリティ対策が必須である。

本発表で報告された Wi-Fi チップセットの場合、当該デバイス上で動作する OS の持つファイルシステムから、起動時にファームウェアを読み込んでいるため、通常のセキュリティ対策も行わなくてはならない。

攻撃の脅威から様々なユーザー、デバイスを保護するためには、今後 Ring3～0 以下のレイヤーにおけるセキュリティ対策手法を研究して行く必要があると考える。

## 3.3. アウトオブバンド管理機能の脆弱性解説

### 3.3.1. 発表の概要

「Intel AMT Stealth Breakthrough」[16] は、Intel AMT（Active Management Technology）と呼ばれる、リモートコンピュータの制御をハードウェアレベルで実装した機能に関する脆弱性についての発表である。

Intel AMT では独自の通信プロトコルが実装されており、電源の ON/OFF から BIOS の設定、OS の起動と言ったソフトウェアの制御まで、KVM（Keyboard, Video, Mouse）を用いて制御する事ができる。

本発表では、コンピュータ情報の確認と電源の制御に用いる Web インターフェースに存在する脆弱性の指摘と、Intel AMT とその基盤である Intel ME（Management Engine）のファームウェア分析についての議論を行っている。

### 3.3.2. Intel AMT と Intel ME

一般によく知られている遠隔操作サポート技術は、ソフトウェア上で実装・実現されている。Windows リモートアシスタンスや VNC（Virtual Network Computing）技術、Chrome Remote Desktop などが典型例である。

これらは OS が起動した後にサービスを開始するため、通常は電源断状態から利用する事はできない。

Intel ME は、CPU や OS から独立したマイクロプロセッサー上で動作するシステムであり、電源状態が S5（コンピュータの電源投入に必要な電力のみが供給されている状態）であっても動作が可能である。

なお、Intel ME は実装されているプラットフォームにより名称とファームウェアが異なり、デスクトップ機器向けは Intel ME、サーバー機器向けは Intel SPS（Server Platform Services）、モバイル機器向けには Intel TXE（Trusted Execution Engine）となっている。

Intel AMT は Intel ME 上で動作するアプリケーションの 1 つであり、ハードウェアレベルの遠隔操作技術の実装である。Intel ME 上で動作するアプリケーションである性質上、システムに対して持つアクセス権限（リングプロテクション）も非常に強力で、ハイパーバイザやシステム管理モードよりも大きな特権を持つ。システム中の各コンポーネントが持つ権限を、図 3-3 に示す。

以上より、Intel ME、Intel AMT に脆弱性が存在した場合、システムの制御をハードウェアレベルで乗っ取られる可能性がある。

Intel AMT は、ローカル環境とリモート環境の双方から接続できる。リモート環境からの遠隔操作技術の実装については、大きく分けて VNC サーバー機能、Web インターフェース機能、SOL（Serial-over-LAN）、IDE-R（IDE-Redirection）、KVM リダイレクト機能を持つ。これらを利用する際にはユーザー認証が必要であり、ダイジェスト認証と Kerberos 認証の 2 つをサポートしている。それぞれの機能が利用するポート番号を、表 3-1 に示す。


**図 3-3 システム権限の階層構造**

| ポート | 用途 |
|---|---|
| 5900 | 暗号化なしのVNCサーバー |
| 16992 | httpを用いたWebインターフェース |
| 16993 | httpsを用いたWebインターフェース |
| 16994 | 暗号化なしのSOL, IDE-R, KVMリダイレクト |
| 16995 | 暗号化ありのSOL, IDE-R, KVMリダイレクト |

**表 3-1 Intel AMT が利用するポート番号と用途**

### 3.3.3. CVE-2017-5689

発表者らは、Intel AMT 使用時のユーザー認証回避による権限昇格が可能な脆弱性を発見し、2017 年 5 月に Intel に対して報告を行った。本脆弱性は CVE-2017-5689 として登録されており、既に修正が行われている。

本脆弱性は、先述の Web インターフェース利用時のダイジェスト認証を回避する事により権限昇格を行っている。使用しているポート番号は 16992 番ポートであり、暗号化なしの http による Web インターフェース機能を対象とする。

発表者らが Intel ME・AMT のファームウェアに対してリバースエンジニアリングを行った結果、一部関数の返り値を改ざんする事で認証回避が可能である事を発見した。即ち、任意のパスワードで認証されるという事である。

先述の通り、Intel AMT は CPU や OS から独立した環境で動作しており、システム中で最高レベルの権限で動作する。また、電源状態や OS に依存しない特徴により、ソフトウェア上に存在する脆弱性よりも影響がはるかに大きい。

オフィスや Kiosk において利用されている一般的なコンピュータだけではなく、産業用コンピュータなどの Intel AMT を実装した様々なシステムが影響を受ける点も特徴である。

攻撃者は、本脆弱性を利用してシステムに侵入する事で、Intel AMT の機能を自由に利用する事ができる。ディスプレイに関する設定を変更する事により、攻撃を完全に秘匿する事も可能である。

### 3.3.4. 対応策

発表者らは、本脆弱性への対応策として、次の方法を挙げている。

- Intel AMT が有効化されていないか、定期的に確認する。
- Intel MEI ドライバを削除する。
- Intel AMT が利用しているポートを、ファイアウォールでブロックする。

本脆弱性は既に修正されているため、最新のファームウェアにアップデートを行っていれば、影響は受けない。

### 3.3.5. 考察

プロセッサやメモリの小型化・高性能化は絶えず進んでおり、従来であれば PC のアプリケーションで実現していた技術が、小型のハードウェアにも実装できるようになっている。

本発表で指摘された Intel AMT はその典型例であり、企業のシステム管理者などが大量のコンピュータを効率的に管理するための強力な機能である。

このような機能の脆弱性が攻撃されると致命的な被害が発生する恐れがあるため、管理するシステムに実装されている機能を把握する事はセキュリティ対策を行う上で非常に重要である。また、実装・有効化されている機能を分かりやすくユーザーに伝える事もベンダーが果たすべき責任の 1 つであると考える。

一般に普及しているシステムにおいて、ハードウェアへ高度な機能を実装する試みは近年の潮流であり、成熟した技術であるとは言い難い。FPGA を始めとするプログラマブルロジックデバイス、その上で動作するシステムを記述する VHDL・Verilog・System-C 等のハードウェア記述言語も年々進化を続けており、高度な機能が容易にハードウェアに実装できるようになってきている。ソフトウェアレベルでのセキュリティ対策だけではなく、ハードウェアレベルでのセキュリティ対策も、重大な課題である。

## 3.4. 機械学習による悪性 URL 検知精度と持続性の検証結果

### 3.4.1. 発表の概要

「Garbage In Garbage Out: How Purportedly Great Machine Learning Models can be Screwed Up by Bad Data」[17]は、アンチウイルス製品に用いられている機械学習を用いたマルウェア検知システムの精度に関する発表である。

昨今のアンチウイルス製品には、機械学習を用いたマルウェア検知システムが搭載される事が増えてきている。

機械学習で生成される学習モデルは、入力したデータに応じて出力を生成するブラックボックスと捉えられ、事前にテストデータセットを用いて入出力のフィッティングを行っている。

そのため最終的な出力精度は、この学習プロセスでのフィッティングに利用するデータに影響を受けている。

本発表では、アンチウイルス製品に実装されている学習モデルの多くは偏りのあるデータセットを用いてフィッティングを行っており、そのためデプロイ後のマルウェア検出精度に問題が生じていると主張している。

### 3.4.2. 機械学習のプロセス

機械学習には多くの学習プロセスが存在するが、本発表で論じているのは教師あり学習についてである。

教師あり学習とは、正解のあるデータセットを入力として用い、学習モデルが出力する結果を正解と比較することで、精度を向上する学習モデル生成手法である。

アンチウイルス製品であれば、事前にマルウェアと判明しているデータを入力することにより、マルウェアの特徴を捉えた学習モデルとなるようフィッティングを行っている。

発表では、よいデータセットの条件、即ち検知精度が一定に保たれる学習モデルを生成するために必要なデータセットの条件として、以下の 2 点を挙げている。

- デプロイ後においても、学習モデルが一貫した性能を発揮する事が出来るデータセットを利用する事。
- 現実に存在するデータセット、実際のマルウェア群を学習用データセットとして用いる事。

### 3.4.3. 異なるデータセットを用いた比較

本発表では、CommonCrawl（PhishTank）、Sophos、VirusTotalと言う３つのマルウェアデータベースを用いて、学習モデルの生成を行っている。

CommonCrawl はフィッシングサイトの URL を蓄積したデータベースで、Sophos は発表者の所属する企業の持つデータベース、VirusTotal はインターネット上に存在するマルウェア情報データベースである。

本発表では、畳み込みニューラルネットワークを用いて、悪性URL 群から学習モデルを生成する手法を用いて生成されている。これは入力に文字列を取り、文字毎に多次元ベクトルを生成、最終的に与えられた URL の不審さの点数を出力するモデルである。発表者らは、この手法で生成する学習モデルを"URL Model"と称している。

学習モデルの正確さの判定には、"AUC（Area Under the [ROC] Curve）"と呼ばれる評価方法を利用している。これは学習モデルの出力した結果の正解率を縦軸、不正解率を横軸に取ったグラフにプロットする。AUC の値が大きいほど、正確な検知を行える学習モデルである事を示す。

### 3.4.4. 生成した学習モデルの評価

それぞれのデータセットを用いて学習モデルを生成した後、自身・別のデータセット全てにおいて総当たり式で検出精度を検証している。検証結果を、図 3-4 に示す。

その結果、VirusTotal のデータセットを用いて生成した学習モデルが、最も汎化性能が高い結果となった。

CommonCrawl、Sophos のデータセットを用いて生成した学習モデルは、自身のデータセットに対する検知精度は非常に優れているものの、他のデータセットに対する検知精度は VirusTotal のデータセットによる学習モデルと比較して劣る結果となった。

また、各モデルに対して、各モデルを生成した翌月に最新のデータセットを入力したところ、検知率が大きく下がる結果となった。

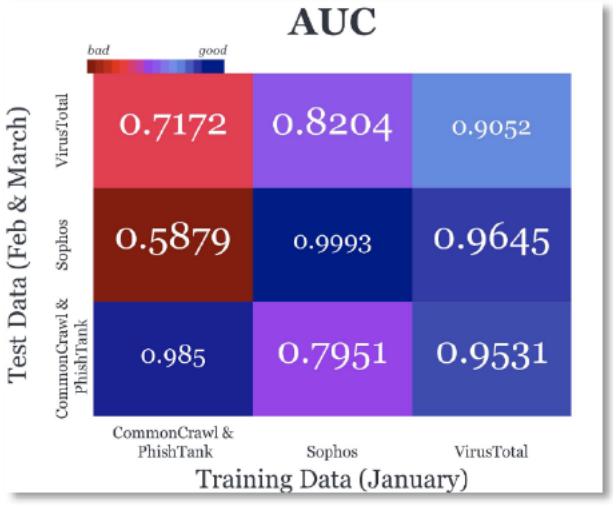


**図 3-4 生成したモデルの評価結果 （出典）発表資料[18]**

### 3.4.5. 考察

VirusTotal のデータセットによる学習モデルが汎化性能に優れる理由は、その性質によるものであると考えられる。VirusTotal はオンラインでマルウェアの情報を蓄積しているデータベースであり、誰でも利用することが出来る。

このことより、VirusTotal のデータセットはいい意味で雑多なデータを蓄積しており、一貫した性能を持つ学習モデルが生成できたものと考えられる。

また、時間経過により検知率が低下する原因は、データセットとして用いた URL の特性によるものと考えられる。

URL は通常のマルウェア群と比較して、その傾向・動向が変化する速度が速い。通常のマルウェアに対する機械学習システムの検知精度は、本発表の URL ベースの機械学習システムほど極端な精度低下は生じないと考えられる。

一方で、マルウェアであっても、学習モデル生成時に利用する特徴量次第では、急速に検知精度が低下する可能性も十分にある。従って、機械学習を用いたマルウェア検知システムを利用しているセキュリティベンダーは、定常的にマルウェアの分析を行い続ける必要がある。

## 3.5. コネクテッドカーの脆弱性解説

### 3.5.1. 発表の概要

「Free-Fall: Hacking Tesla from Wireless to CAN Bus」[19]は、2016 年 9 月に発表された、Tesla Model S の脆弱性に関する発表の続報である。複数の脆弱性を連鎖させる事により、遠隔からの乗っ取り攻撃が可能である事を示した。

以下に、発表された具体的な脆弱性の情報と、それを用いた攻撃手法について説明する。なお、本脆弱性と攻撃手法は既に Tesla へ報告済みであり、修正が行われている。

### 3.5.2. 攻撃手法

本発表で用いられた脆弱性は、次の通りである。

- 無線 LAN ネットワークにおける、脆弱な認証情報の利用

Tesla Model S は車両側に無線 LAN 子機を持っており、Teslaの販売店や充電スタンド（Supercharger）に存在するホットスポットに接続し、通信を行っている。

この際、利用しているホットスポットの SSID、パスワードがそれぞれ、"Tesla Guest"・"abcd123456"であり、非常に脆弱なものであった。

上記 SSID・パスワードを設定したホットスポットを用意することにより、車両側からアクセスさせることが可能である。

- IVI システムへの脆弱性攻撃

次に、Tesla Model S の IVI（In-Vehicle Infotainment）には、既知の脆弱性が存在する古いバージョンのコンポーネントが実装されていた。

先に述べたホットスポットへの接続後、ブラウザを不正なサイトに誘導し、ブラウザへの脆弱性攻撃によりシステムへ侵入する。

システムへ侵入した段階では、攻撃者が持つシステム権限は Web ブラウザを実行しているユーザーの権限となる為、車両の CAN ネットワークへのアクセス権限が不足しており、権限昇格が必要である。

Tesla Model S では、IVI システムにおいて Linux Kernel 2.6.36 を使用していた。これも先の Web ブラウザ同様、既知の脆弱性が存在する古いバージョンである。

システムカーネルに対しても同様に脆弱性攻撃を行い、攻撃者は IVI 中のルート権限の取得と、AppArmor の無効化を行い、制御系ネットワークの入り口となる CAN ゲートウェイへのアクセスを可能にする。

- ファームウェアアップデート機能の悪用

IVI 側から車内 CAN ネットワークへ制御情報を送信するためには、CAN ゲートウェイを経由する必要がある。

CAN ネットワークに対して任意の CAN メッセージを送信するためには、CAN ゲートウェイのファームウェアを書き換える必要があった。発表者らは、改ざんしたアップデータを用いて、CAN ゲートウェイのファームウェアを強制的に上書きすることで、CAN ネットワークへの接続を実現している。

これらの脆弱性を連鎖させる事で、物理的アクセスのない状態から車両への侵入が可能であったとのことである。連鎖の様子を、図 3-5 に示す。

なお、本脆弱性について発表者らが Tesla 社へ報告後、10 日で修正プログラムが配布された。

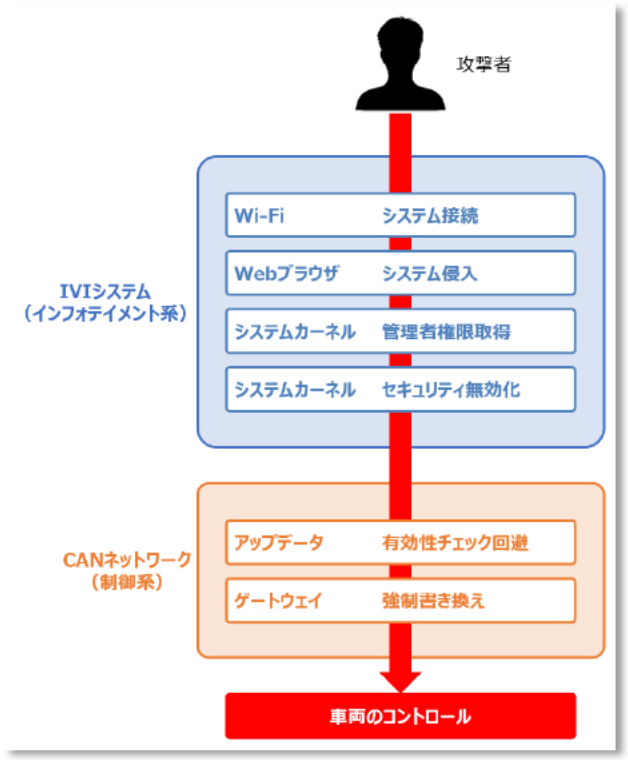


**図 3-5 脆弱性攻撃の連鎖**

### 3.5.3. 考察

車両システムに存在する複数の脆弱性を連鎖させる事で、本発表にあった遠隔乗っ取り攻撃は実現されている。攻撃に利用された脆弱な認証情報の利用、Web ブラウザと Linux カーネルにおける既知の脆弱性、回避が容易なファームウェアのコード署名のうち、いずれかが対策されていれば攻撃は不可能であったと考えられる。

AUTOSAR や AGL（Automotive Grade Linux）の採用により車載システムを標準化することで、セキュリティとユーザビリティの向上、開発コストの圧縮などが期待できる。これらを採用することで車載システムの標準化も進み、より高品質なシステムが構築しやすくなると考えられる。AUTOSAR の理念にもある通り、標準化においては、業界各社で協力が必須である。

また、認証情報やファームウェアのコード署名についても、製品開発ベンダーで適切なポリシーを設定し、顧客と製品を守る事を最優先としなくてはならない。

製品開発ベンダーは安全機能のみではなく、セキュリティ機能に対しても責任を持つ必要がある。

# 4. 参考文献


1. 日本工業標準調査会, "JIS Q27002：2006（ISO/IEC 17799：2005）情報技術−セキュリティ技術−情報セキュリティマネジメントの実践のための規範", <http://kikakurui.com/q/Q27002-2006-01.html>, 2006
2. Adam Shostack, "About Threat Modeling: Designing for Security", <https://threatmodelingbook.com/>, 2014/02/17
3. 松並 勝, 脅威分析研究会/SIGSTA,"開発現場で実践している脅威分析(セキュリティ設計分析)事例紹介", <https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxzaWdzdGF3ZWJ8Z3g6N2MwZDgyZGExYmY1YzFhYg>, 2016/1/12
4. "Microsoft", "Microsoft Threat Modeling Tool 2016", <https://www.microsoft.com/en-us/download/details.aspx?id=49168>
5. NPO 日本ネットワークセキュリティ協会セキュリティ学調査ワーキンググループ, "2014 年情報セキュリティインシデントに関する調査報告書～個人情報漏えい編～第 1.0 版", < http://www.jnsa.org/result/incident/data/2014incident_survey_ver1.1.pdf> ページ 56, 2016/6/16
6. 脅威分析研究会 / SIGSTA,<https://sites.google.com/site/sigstaweb/about>
7. Black Hat Asia 2017, <https://www.blackhat.com/asia-17/>, 2017/4
8. Bing Sun et al., McAfee LLC, " The Power of Data-Oriented Attacks: Bypassing Memory Mitigation Using Data-Only Exploitation Technique Part I", < https://www.blackhat.com/docs/asia-17/materials/asia-17-Sun-The-Power-Of-Data-Oriented-Attacks-Bypassing-Memory-Mitigation-Using-Data-Only-Exploitation-Technique.pdf>, 2017/4
9. Hong Hu et al., Department of Computer Science, National University of Singapore, "Automatic Generation of Data-Oriented Exploits ", <http://www.comp.nus.edu.sg/~prateeks/papers/DOP.pdf>, 2015/8
10. Bing Sun et al., McAfee LLC," The Power of Data-Oriented Attacks: Bypassing Memory Mitigation Using Data-Only Exploitation Technique Part II ", < https://conference.hitb.org/hitbsecconf2017ams/materials/D2T1%20-%20Bing%20Sun%20and%20Chong%20Xu%20-%20Bypassing%20Memory%20Mitigation%20Using%20Data-Only%20Exploitation%20Techniques.pdf >, 2017/4
11. Monnappa K A, Cisco Systems G.K, " What Malware Authors Don't Want You to Know - Evasive Hollow Process Injection", < https://www.blackhat.com/docs/asia-17/materials/asia-17-KA-What-Malware-Authors-Don't-Want-You-To-Know-Evasive-Hollow-Process-Injection-wp.pdf>, 2017/4
12. Tongbo Luo et al., Palo Alto Networks, Inc., "ANTI-PLUGIN: DON'T LET YOUR APP PLAY AS AN ANDROID PLUGIN", < https://www.blackhat.com/docs/asia-17/materials/asia-17-Luo-Anti-Plugin-Don't-Let-Your-App-Play-As-An-Android-Plugin-wp.pdf>, 2017/4
13. Black Hat USA 2017, <https://www.blackhat.com/us-17/>, 2017/7
14. Andrea Continella et al., Politecnico di Milano and Trend Micro Inc., "ShieldFS: The Last Word In Ransomware Resilient Filesystems", <https://www.blackhat.com/docs/us-17/wednesday/us-17-Continella-ShieldFS-The-Last-Word-In-Ransomware-Resilient-Filesystems-wp.pdf>, 2017/7
15. Nitay Artenstein et al., Exodus Intelligence, "Broadpwn: Remotely Compromising Android and iOS via a Bug in Broadcom's Wi-Fi Chipsets", <https://www.blackhat.com/docs/us-17/thursday/us-17-Artenstein-Broadpwn-Remotely-Compromising-Android-And-iOS-Via-A-Bug-In-Broadcoms-Wifi-

Chipsets-wp.pdf>, 2017/7
16. Dmitriy Evdokimov et al., Embedi, "Intel AMT Stealth Breakthrough", <https://www.blackhat.com/docs/us-17/thursday/us-17-Evdokimov-Intel-AMT-Stealth-Breakthrough-wp.pdf>, 2017/7
17. Hillary Sanders, and Joshua Saxe, SOPHOS Ltd., "GARBAGE IN, GARBAGE OUT: HOW PURPORTEDLY GREAT ML MODELS CAN BE SCREWED UP BY BAD DATA", <https://www.blackhat.com/docs/us-17/wednesday/us-17-Sanders-Garbage-In-Garbage-Out-How-Purportedly-Great-ML-Models-Can-Be-Screwed-Up-By-Bad-Data-wp.pdf>, 2017/07
18. Hillary Sanders, and Joshua Saxe, SOPHOS Ltd., "GARBAGE IN, GARBAGE OUT: HOW PURPORTEDLY GREAT ML MODELS CAN BE SCREWED UP BY BAD DATA", < https://www.blackhat.com/docs/us-17/wednesday/us-17-Sanders-Garbage-In-Garbage-Out-How-Purportedly-Great-ML-Models-Can-Be-Screwed-Up-By-Bad-Data.pdf>, 2017/07
19. Sen Nie, Ling Liu, and Yuefeng Du, Keen Security Lab of Tencent, "FREE-FALL: HACKING TESLA FROM WIRELESS TO CAN BUS", <https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf>, 2017/07

**株式会社 FFRI について**

FFRI は、日本発のサイバーセキュリティをリードする専門家集団です。

国際的なセキュリティカンファレンスでの研究発表実績もある世界トップレベルのサイバーセキュリティ専門家集団が、 先進的な調査結果により、今後予想される脅威を先読みし、一歩先行くコンセプトで製品・サービスを展開しています。



株式会社 FFRI

〒150-0013 東京都渋谷区恵比寿 1 丁目 18 番 18 号 東急不動産恵比寿ビル 4 階

E-mail: research-feedback [at] ffri.jp URL:http://www.ffri.jp/