

15-A-9

ソニーの電子お薬手帳システム「harmo」に適用した セキュリティ設計分析手法¹

1. 概要

本編では、ソフトウェアの仕様・設計に対するセキュリティ分析技術（セキュリティ設計分析）の実製品への適用事例として、ソニー株式会社の電子お薬手帳システム「harmo（ハルモ）」（以下「電子お薬手帳システム」とする）の例を紹介する。

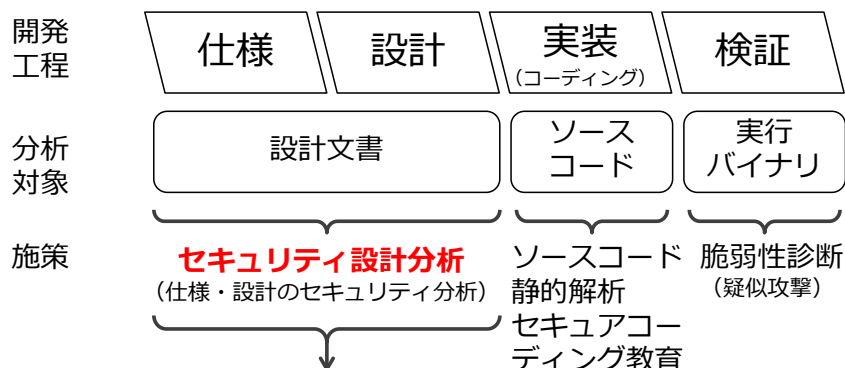
インターネットとコンピュータシステムが重要な社会基盤となった今、セキュアなコンピュータシステムの開発方法の重要性が高まってきた。特に医療系のコンピュータシステムでは患者の医療情報を扱うため、高度なセキュリティが求められる。今回、事例として紹介する電子お薬手帳システムには、薬剤師が患者の薬歴情報と処方箋から危険な薬の組み合わせを確認する医療業務支援機能がある。ここで患者の薬歴情報が第三者に改ざんされていると危険な薬の組み合わせを発見できない危険性がある。また患者の薬歴情報が漏えいすると、患者が重病を抱えていることを他人に知られてしまうなどの危険性もある。このような問題を起こさないために、電子お薬手帳システムには高度なセキュリティが求められる。

セキュアなコンピュータシステムを作るためには、まず(a)セキュアな仕様・設計を作り、それに基づき(b)セキュアなソースコードを作り、(c)セキュリティ観点でシステムを検査する必要がある。(b)および(c)には静的解析ツールや動的検査ツールなどの自動化された解決策やセキュアコーディング教育があり、広く適用され普及している。一方、(a)については自動化されたものではなく、人手による高度な知識と経験を要する仕様・設計のセキュリティ分析（以下、セキュリティ設計分析とする）しかなく、セキュアな仕様・設計作りはまったくと言ってよいほど普及していない。しかしながらその重要性は益々高まっている。

図 15-A-9-1 で示すように、セキュリティ設計分析では、暗号技術や OS 毎の癖などの個別技術のセキュリティ知識と、それら知識を基礎としシステム全体のセキュリティ状況を可視化する分析技術の 2 つのスキルが必要となる。本編では後者の可視化技術について従来の課題を解決した分析手法を事例として紹介する。

¹ 事例提供: 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 松並 勝 氏

• セキュリティ設計分析の位置づけ



• セキュリティ設計分析に求められるスキル

1. 暗号やOSなどの個別技術のセキュリティ知識

**2. それら知識を基礎とした
システム全体のセキュリティ状況の可視化技術**

← 本稿のターゲット

図 15-A-9-1 本編で事例紹介する分析手法の位置づけ

2. 課題

セキュリティ設計分析の代表的な手法の一つに脅威モデリング（または脅威モデル）がある。脅威モデリングはマイクロソフト製品のソフトウェア開発で実践²されており、近年のマイクロソフト製品のセキュリティ品質の高さにより脅威モデリングの効果は実証されている。

脅威モデリングでは図 15-A-9-2 のような脅威ツリーという FTA（Fault Tree Analysis）に似た分析木を作成する。脅威ツリーの各最上位ノードには脅威（望ましくない事象）を記載し、その下に脅威の顕在化条件（脅威が生じるための必要条件）を繰り返し分解しツリーを形成する。最上位ノードに記載する脅威とは攻撃者が目的とするコンピュータシステム上で発生させたい事象の記述である。

² マイクロソフト社は Security Development Lifecycle と呼ばれるソフトウェアセキュリティプラクティス群を自ら実践しているだけでなく、Web サイトで一般公開している。脅威モデリングはその中の 1 つのプラクティスである。 <http://www.microsoft.com/security/sdl/default.aspx>

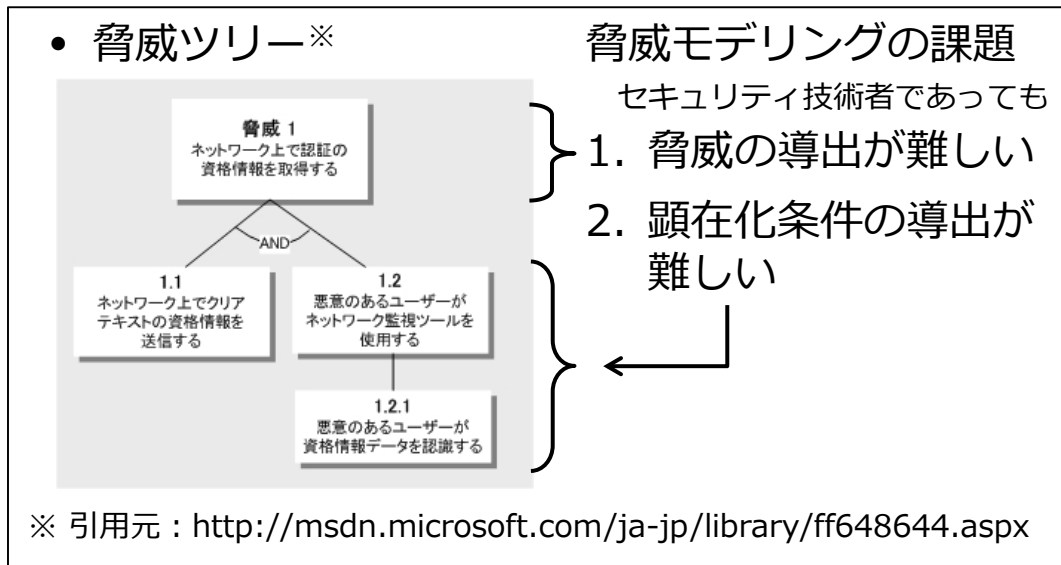


図 15-A-9-2 脅威ツリーと脅威モデリングの課題

脅威ツリーの作成は、暗号技術や OS 毎の癖などの個別技術のセキュリティ知識が豊富なセキュリティ技術者であっても難しい。具体的には次の 2 つの難しさがある。

- (1) 脅威（望ましくない事象）を洗い出すことが難しい
- (2) 脅威の顕在化条件を構築することが難しい

脅威モデリングを普及させるためには、この 2 つの難しさを解決する必要がある。これらの難しさはそれぞれの分析手順が確立していないことに起因する。本編ではこの 2 点について分析手順をパターン化することで難しさを解消する改善策を提案する。

3. セキュリティ設計分析の基礎知識

本編提案の主題である分析手順のパターン化の議論に入る前に、その基礎となる概念を説明する。ここで説明する概念は提案手法における考え方であり、必ずしも一般的な考え方とは言えない部分もあるのでご留意いただきたい。

3.1. セキュリティ要件

3.1.1. 脅威とセキュリティ要件

脅威モデリングにおいては「脅威」を最上位ノードとするツリーを作成するが、提案方法では「セキュリティ要件」を最上位ノードとするツリーを作成する。脅威は FTA と同様に望ましくない事象の表現であるが、セキュリティ要件はコンピュータシステムに期待する望ましい状態や性質の表現である。脅威とセキュリティ要件は表 15-A-9-1 に示した例のように、意味的には同じ概念を攻撃者の立場、防御の立場で表現しただけの違いであり、本質的には同じものであると考えられる。脅威とセキュリティ要件のどちらで表現してもよいのである。

なら、ソフトウェア工学の既存概念である要件（要求）に揃えてセキュリティを表現するほうがソフトウェア開発者に馴染みがあり望ましいと考える³。

表 15-A-9-1 脅威とセキュリティ要件

脅威	第三者が患者の薬歴情報を参照できる（できてしまう）
セキュリティ要件	第三者は患者の薬歴情報を参照できない（できてはならない）

3.1.2. セキュリティ要件とは

セキュリティ要件とはコンピュータシステムのステークホルダがそのコンピュータシステムに期待するセキュリティ観点の性質、つまり脅威（望ましくない事象）を発生させないためにコンピュータシステムに求められる性質のことである。表 15-A-9-1 に示したセキュリティ要件の例は「第三者は患者の薬歴情報を参照できない（できてはならない）」という性質をステークホルダがシステムに期待していることを表現している。

コンピュータシステムの「セキュリティ」は次のように抽象的に捉えると分かりやすい。コンピュータシステム上で生じる様々な事象について、ステークホルダである人間がそれぞれの立場において、その事象は生じて欲しくない、または逆に、たとえ生じたとしても特に困らない、などと考えるのである。コンピュータシステムにとってはどれも単なる事象に過ぎないが、そのステークホルダである人間が勝手に「その事象は生じて欲しくない」とか「その事象は生じて構わない」と言っているに過ぎない。

また多くの場合、事象を発生させるのが「誰であるか」によって、事象に対する善し悪しの判断が変わる。たとえば、患者の薬歴情報はその患者自身であれば読めても構わないが、第三者に読まれてしまうことは許されない。このように「ユーザーが薬歴情報を参照する」というコンピュータシステム上で生じる「事象」を誰が生じさせているのかによって善し悪しの判断が変わるのも、コンピュータシステムのセキュリティの特徴である。

以上のことから、セキュリティ要件は「ステークホルダである人間の主観や価値観により定まる、生じてよい事象、生じてはならない事象を何らかの表現形で表したもの」であると言える。

主観や価値観は人により異なるうえ時代とともに変遷するものである。そのため個々のコンピュータシステムから独立した普遍的なセキュリティ要件を一律に定めることは難しい。よって個々のコンピュータシステムにそのステークホルダが合意するセキュリティ要件を都度記述し、表現する必要があることに注意すべきである。

³ 一方、FTA は望ましくない事象であるハザードを起点にした分析方法であり、脅威モデリングは FTA と同じ立場をとっている。FTA がセーフティ分野で十分確立した手法であることから、本編で主張しているような、脅威ではなくセキュリティ要件を起点にするセキュリティの捉え方に価値があるのかどうかは、今後の事例の積み重ねで確認していかなければならない。

3.1.3. セキュリティ要件の表現方法

前節でセキュリティ要件とは、コンピュータシステム上で生じる「事象」に関して、生じてよい事象、生じてはならない事象を何らかの表現形で表したものであると述べた。

コンピュータシステム上で生じる「事象」の表現方法や表現粒度は様々である。そのため脅威モデリングにおいて脅威（望ましくない事象）の表現方法や表現粒度が人によってまちまちとなる課題があった。本編の提案方法においてもセキュリティ要件の表現手法や表現粒度について同じ課題があった。

そこでセキュリティ要件の表現方法と表現粒度に何らかの指針を設けるために次のように考えた。コンピュータシステム上で生じる「事象」は、突き詰めていくと、最終的には「情報」の読み（READ）、書き（WRITE）、そして「機能」の実行（EXECUTE）という単純な操作（オペレーション）で構成されていると考えることができる。さらにこうした「事象」を「誰が」生じさせているかによってセキュリティ観点の判断が決まることも踏まえ、コンピュータシステム上で生じる「事象」を図 15-A-9-3 の表現パターンで表現することにした。つまり事象をアクター、資産、操作の 3 要素で構成する。

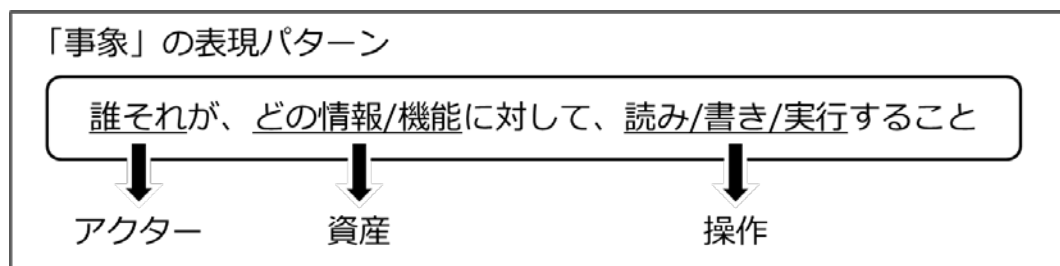


図 15-A-9-3 「事象」の表現パターン

前述の情報と機能をまとめて「資産」と呼ぶ。資産とは要するに「守るべきもの」である。平たく言えば、資産とは勝手に使われては困るような情報や機能のことであり、それぞれ「情報資産」、「機能資産」とも呼ぶ。「勝手に使われて」と書いたが、「使われて」の部分資産に対する「操作」である。情報資産については通常、読み（READ）と書き（WRITE）の操作があり、機能資産については実行（EXECUTE）という操作があることは前述の通りである⁴。

図 15-A-9-3 における「アクター」とは対象となるコンピュータシステムに関与しうる登場人物である。当該システムのステークホルダはもちろんのこと、攻撃者となりうる人物も含む。ここで登場人物は攻撃者や被害者といった区別をせず単に「アクター」と呼ぶ。一般に被害者（保護対象の人物）として考えられる正規の利用者であっても、ときにシステムを悪

⁴ 資産によっては READ、WRITE、EXECUTE といった操作で表現するのでは粒度が粗くて不十分な場合もある。その場合、資産に合わせて適切な操作を定義するとよい。たとえば情報資産の操作を READ、WRITE ではなく、CREATE、READ、UPDATE、DELETE で表現することで、CREATE と DELETE の許可・不許可の要件を UPDATE の要件と区別することができるようになる。

用する場合もあり、その場合は正規の利用者も攻撃者となるからである。電子お薬手帳システムでは表 15-A-9-2 の 4 種類のアクターを想定した。

表 15-A-9-2 電子お薬手帳システムで想定したアクターの例

アクター	アクターの説明
患者	当システムを利用している患者
薬剤師	当システムを導入した薬局の薬剤師
ソニー	当システムの運用にあたる特別な権限を有する特定少人数
第三者	上記のいずれでもない人

アクターを分類するとき、細かく多数のアクターに分類してしまうと、分析のときの組み合わせ空間が巨大になってしまうので、適度な分類に抑えておく必要がある。筆者の経験としては、システムへのアクセス権限の違いに合わせてアクターを分類するのが、ほどよいアクター分類を導くのに適していると感じている。

3.2. 特殊な命題の解釈

提案手法のノードに記載する命題の解釈方法には、少し特殊な解釈の仕方があるので説明する。

3.2.1. 暗号化された情報の READ、WRITE

とある「〇〇情報」が暗号化されてファイル保存されている場合を想定する。このとき、攻撃者がそのファイルの内容を **READ** できたとしても、もともとの〇〇情報の内容（つまり平文の〇〇情報）は **READ** できるとは考えない。逆に、攻撃者がそのファイルの内容を **READ** できて、なおかつ攻撃者がその暗号鍵（の平文）を **READ** できるときに、攻撃者は〇〇情報を **READ** できると考える。また、攻撃者が〇〇情報の内容を何らかの方法で推測できる場合も、攻撃者は〇〇情報を **READ** できると考える。

次に「〇〇情報」が電子署名されてファイル保存されている場合を想定する。このとき、攻撃者がそのファイルの内容を **WRITE** できたとしても、〇〇情報を **WRITE** できるとは考えない。電子署名により改ざんが検知され **WRITE** された値が使われないからである⁵。もし攻撃者が電子署名のための署名鍵（の平文）を **READ** できて、なおかつ攻撃者が〇〇情報を **WRITE** した上で正しく電子署名できた場合、攻撃者は「〇〇情報」を **WRITE** できたと考える。

3.2.2. 資産の裏に隠れる資産

機能はコンピュータシステム上でプログラムコードにより実現されている。プログラムコ

⁵ 攻撃者が〇〇情報の内容を知ることにはできないが、正規の利用者が〇〇情報を参照することを「妨害」することはできる。「妨害」については後述の表 15-A-9-3 セキュリティ要件ひな形で扱っている。

ードの中にはその機能を実現するためのアルゴリズムや鍵、暗号エンジンなど、秘密の情報や機能が含まれることがある。こうした秘密の情報や機能がある場合、それらもまた保護対象であるので資産として扱うことになる。

資産を洗い出すときには「資産の裏にはそれを支える隠れた資産があるかもしれない」と意識する必要がある。

4. 提案

本章では、図 15-A-9-4 に示したセキュリティ要件の導出手順と分析木の分解手順をそれぞれ「4.1 セキュリティ要件の導出手順のパターン化」と「4.2 分析木の分解手順のパターン化」で説明する。

2種類のパターン化した手順を提案する

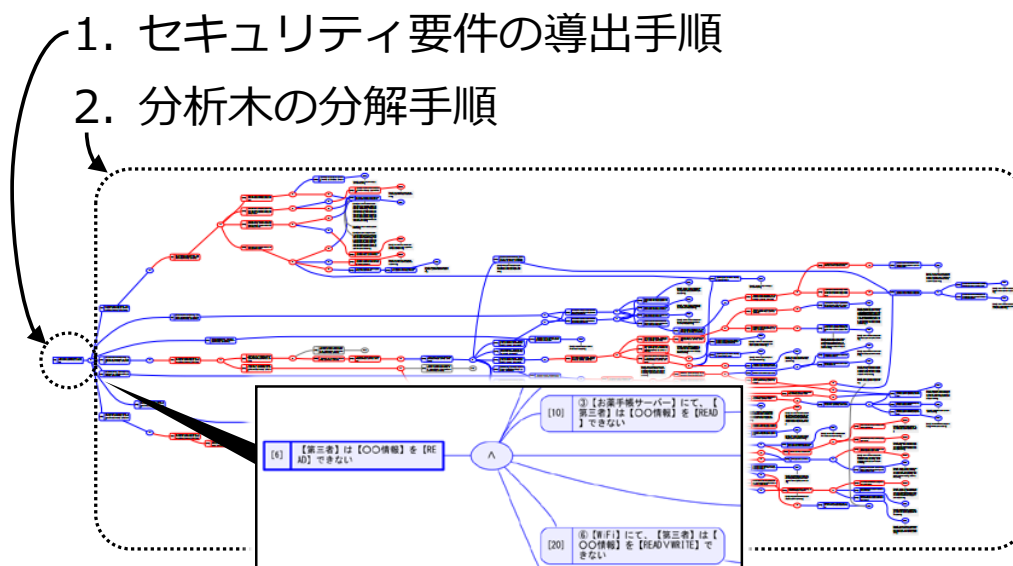


図 15-A-9-4 提案する 2 種類のパターン化した分析手順

提案方法ではグラフのツリーは横向き⁶であり、左端の最上位ノードは脅威ではなくセキュリティ要件が記載され、その右側に記載される下位ノード群は最上位ノードのセキュリティ要件が成立するために必要な条件群が記載されている。また各ノードには命題が記述されている。

⁶ 高解像度ワイドディスプレイでの分析作業には横向きツリーが便利。

本編では説明しないが、分析作業と作図を効率化するために、実際には図 15-A-9-5 のように Excel 上で分析した論理構造を記述し、VBA マクロと Graphviz⁷を使ってグラフ出力している。

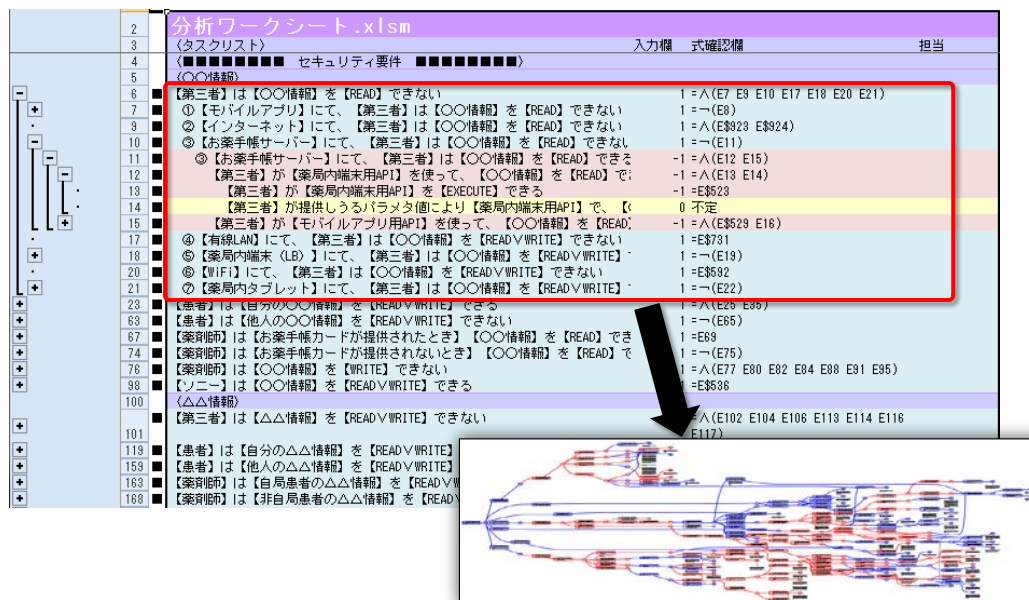


図 15-A-9-5 Excelで論理構造を記述、マクロと Graphviz でグラフ出力

4.1. セキュリティ要件の導出手順のパターン化

基本的なセキュリティ要件の導出手順は次の2段階である。

- (1) 資産を洗い出す
- (2) セキュリティ要件ひな形で要件を作る

4.1.1. 資産を洗い出す

設計文書や開発者ヒアリングを通して、「意図しない人に勝手に使われては困る情報や機能」を見つけることで情報資産や機能資産を洗い出す。3.2.2.で説明した資産の裏に隠れる資産についても注意する。

資産洗い出しの精度や再現性を高めるために、資産となることが多いキーワード（例：クレジットカード番号）をデータベース化し、属人性を減らすなどの工夫も必要である⁸。

4.1.2. セキュリティ要件ひな形で要件を作る

資産に生じる事象はコンピュータシステム上では操作として表現される。情報には READ、WRITE の操作があり、機能には EXECUTE の操作がある。資産に対してどの操作が誰に許

⁷ Graph Visualization Software <http://www.graphviz.org/>

⁸ 資産を洗い出す方法についてはもっと改善の余地がある。まだまだ工夫が足りない。

されて、誰に許されないのかを表現したものがセキュリティ要件である。表 15-A-9-3 のセキュリティ要件ひな形に資産を種別毎に【情報資産】または【機能資産】に当てはめ、アクターを【攻撃者】、【被害者】に当てはめることで、セキュリティ要件の文をパターン的に導出できる。分析対象システムにおいて相応しい文だけをセキュリティ要件として採用することで、当該システムのセキュリティ要件を導出できる。

表 15-A-9-3 セキュリティ要件ひな形

資産種別	情報		機能
操作	READ	WRITE	EXECUTE
セキュリティ要件	【攻撃者】は【情報資産】を【READ】できない	【攻撃者】は【情報資産】を【WRITE】できない	【攻撃者】は【機能資産】を【EXECUTE】できない
	【被害者】が【情報資産】を【READ】することを【攻撃者】が妨害できない	【被害者】が【情報資産】を【WRITE】することを【攻撃者】が妨害できない	【被害者】が【機能資産】を【EXECUTE】することを【攻撃者】が妨害できない

以上の手順によって複数のセキュリティ要件が命題として書き出されることになる。書き出されたすべての命題が TRUE になるようにシステムを設計・実装すれば、システムはセキュアに作られたということになる。図 15-A-9-6 に電子お薬手帳システムのセキュリティ要件の例を示す。

資産	セキュリティ要件	
〇〇情報	【第三者】は【〇〇情報】を【READ】できない	それぞれが命題で表現されている。
ト〇〇X情報	【第三者】は【〇〇情報】を【WRITE】できない	
ト〇〇Y情報	【患者】は【他人の〇〇情報】を【READ】できない	
ト〇〇Z情報	【患者】は【他人の〇〇情報】を【WRITE】できない	
	【薬剤師】は【お薬手帳カードが提供されないとき】【〇〇情報】を【READ】できない	すべての命題が常に成立すれば、このシステムはセキュアであるということ。
	【薬剤師】は【〇〇情報】を【WRITE】できない	
△△情報	【第三者】は【△△情報】を【READ】できない	
	【第三者】は【△△情報】を【WRITE】できない	
	【患者】は【他人の△△情報】を【READ】できない	
	【患者】は【他人の△△情報】を【WRITE】できない	
	【薬剤師】は【非自局患者の△△情報】を【READ】できない	
	【薬剤師】は【非自局患者の△△情報】を【WRITE】できない	
□□情報	【第三者】は【□□情報】を【READ】できない	
ト□□X情報	【第三者】は【□□情報】を【WRITE】できない	
ト□□Y情報	【患者】は【自分の□□情報】を【WRITE】できない	
ト□□Z情報	【患者】は【他人の□□情報】を【READ】できない	
	【患者】は【他人の□□情報】を【WRITE】できない	
	【薬剤師】は【カードが提供されないときは】【他局発行の自局患者の□□情報】を【READ】できない	
	【薬剤師】は【他局発行の自局患者の□□情報】を【WRITE】できない	
	【薬剤師】は【他局発行の非自局患者の□□情報】を【READ】できない	
	【薬剤師】は【他局発行の非自局患者の□□情報】を【WRITE】できない	

※発表用に加工済み

図 15-A-9-6 電子お薬手帳システムにおけるセキュリティ要件一覧の例

4.2. 分析木の分解手順のパターン化

前節まででセキュリティ要件を複数の命題で記述する方法を説明した。ここではセキュリティ要件の一つ一つについて、分析対象のシステムが実際にセキュリティ要件を満たしているかを確認、可視化する分析木の分解手順について説明する。

可視化によりシステムがセキュリティ要件を満たさないことが判明すると、なぜセキュリティ要件を満たせていないのかも同時に分かり、防御策の立案に役立つ。これは、分析木作成の大きなメリットである。

分析木の基本的な作成手順を図 15-A-9-7 に示す。まず一つのセキュリティ要件を選び、その命題を起点ノードとして開始する（手順 1）。次に起点ノードを分解する（手順 2）が、その方法には、分析対象システムの設計知識に基づいて分解する方法、セキュリティ知識（防御方法、攻撃方法）に基づいて分解する方法、そして本編での改善提案である分解パターンに基づいて分解する方法の 3 つがあり、いずれかを選択して実施することで起点ノードを分解する。そして分解して作られた各子ノードにおいて、TRUE または FALSE が判定できない子ノードに対して手順 2 を繰り返す（手順 3）。

- **基本手順（セキュリティ要件ごとに実施）**
 1. セキュリティ要件を起点ノードにして開始
 2. いずれかの方法でノードを分解

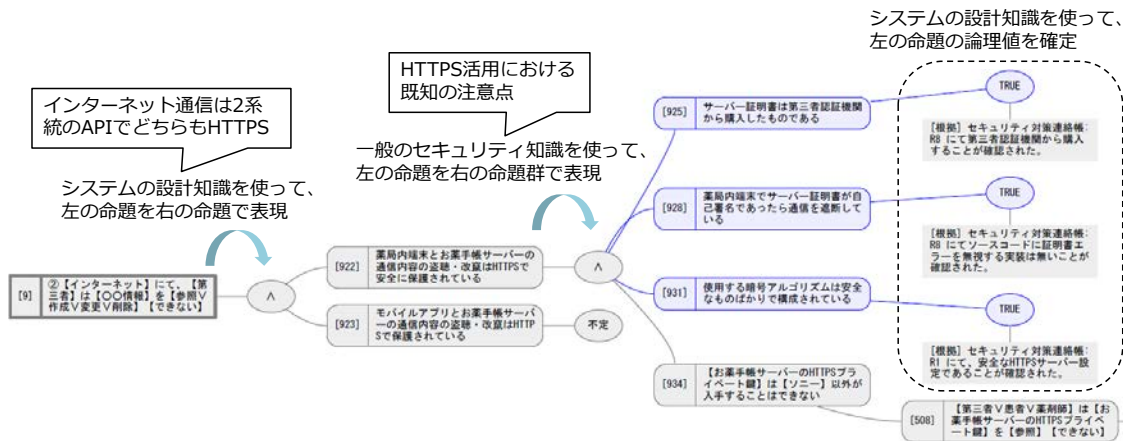
- 分析対象システムの**設計知識**に基づき分解
 - **セキュリティ知識**（防御方法、攻撃方法）に基づき分解
 - **分解パターン**に基づき分解 ← 本稿での改善提案
 3. TRUEまたはFALSEが判定できないノードに対し手順2.を適用 … 繰り返す

図 15-A-9-7 基本的な分析木の分解手順

図 15-A-9-7 の手順 2 の枠内の手順について、図 15-A-9-8 を使って詳しく説明する。図 15-A-9-8 は、分析対象システムの設計知識や一般的なセキュリティ知識を使った分解の様子を表している。例えば分析対象システムの設計知識「インターネット通信は 2 系統の API どちらも HTTPS⁹ で設計されている」を使って、ノード 9 の命題をノード 922, 923 の 2 つの命題で表現している。次に、一般的なセキュリティ知識「HTTPS 活用における既知の注意点」である 4 つの注意点を使って、ノード 922 の命題をノード 925, 928, 931, 934 の命題で表現している。そして、ノード 925 の「サーバー証明書は第三者認証機関から購入したものである」という命題に対して、「セキュリティ対策連絡帳:R8 にて第三者認証機関から購入することが確認された。」という分析対象システムの設計知識を根拠として、ノード 925 は TRUE であると確定している。ちなみに「セキュリティ対策連絡帳」とは、セキュリティ設計分析担当者とシステム設計者との間でやり取りする連絡帳である。セキュリティ設計分析担当者は分析木の作成により明らかになった脆弱性を指摘したり、脆弱性を修正するための設計改善を提案したりする。システム設計担当者は指摘や提案をどのように設計に反映させ

⁹ Hypertext Transfer Protocol Secure

たのかを記録する。また、セキュリティ設計分析担当者はその設計反映方法の妥当性を確認する。



- ノードの意味を維持したまま、システム設計知識や一般的なセキュリティ知識を使って、別の式で表現すること繰り返す
- この分解にパターン化できるものがある → **分解パターン**

図 15-A-9-8 分析対象システムの設計知識や一般的なセキュリティ知識を使った分解

このようにノードの分解手順とは、ノードの命題の意味を維持したまま、分析対象システムの設計知識や一般的なセキュリティ知識を使って、別の命題を組み合わせた式で別表現をつくる、ということを繰り返すことである。

ここで、この分解手順にパターン化できるものがある、ということが本編の提案するところであり、いくつかの分解パターンをこのあと紹介していく。

4.2.1. 分解パターン【情報 READ 防御視点】

<p>【攻撃者】は【情報資産】を【READ】できない</p> <p>→ 【情報資産】が存在し得る場所で\wedge展開する</p> <ul style="list-style-type: none"> すべての防御が成立したとき親ノードの命題が TRUE となるので論理積\wedgeで子ノードを結ぶ
--

これは「誰それが〇〇情報を **READ** できない」と命題記述のあるノードの分解に使える分解パターンである。矢印(→)の行に分解手順が記載されており、この手順に従えばノードを分解できる。ここでは「**【情報資産】が存在し得る場所で \wedge 展開する**」とある。

情報資産はシステム内のあちこちに伝達する。たとえばお薬手帳システムのとある情報資産「〇〇情報」は図 15-A-9-9 のように①モバイルアプリ、②インターネット、③お薬手帳サーバー、薬局内の④有線 LAN、⑤薬局内端末、⑥Wi-Fi 通信路、そして⑦薬局内タブレットといった場所に伝達する。

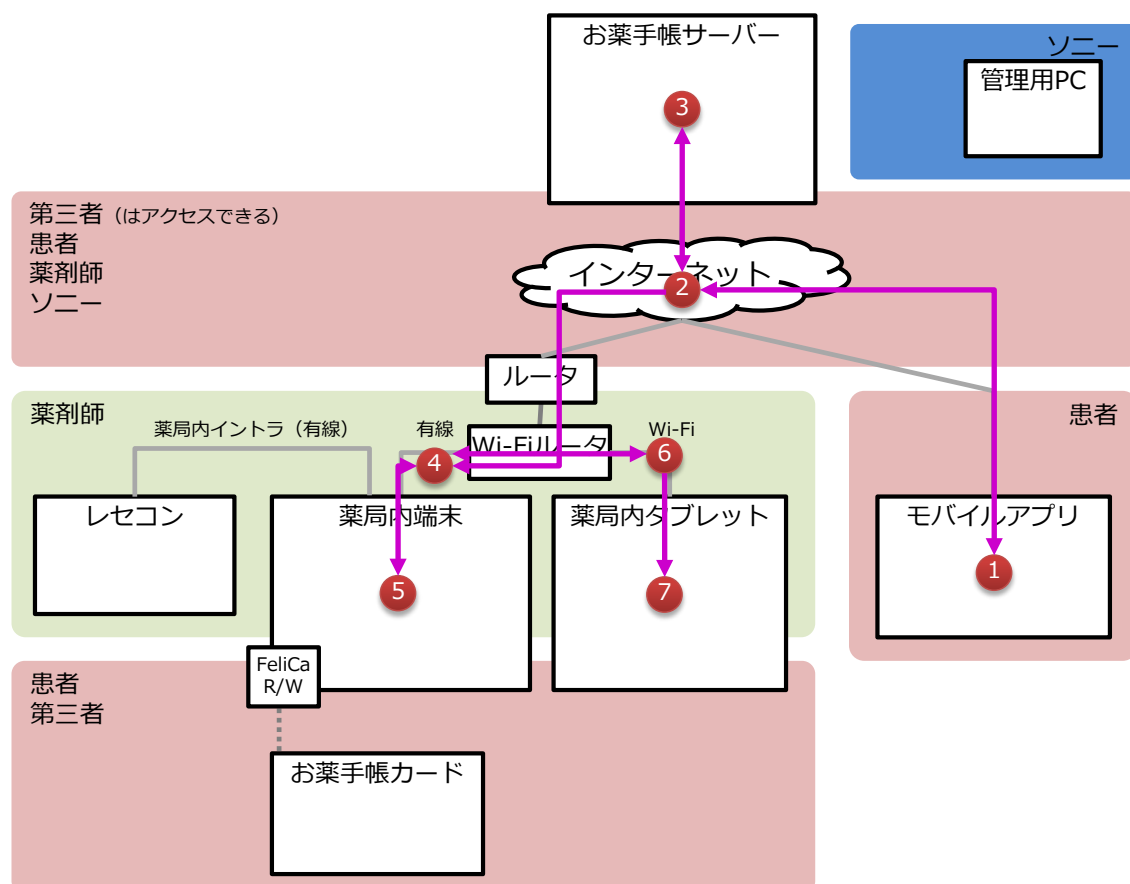


図 15-A-9-9 〇〇情報が存在し得る場所を洗い出したデータフロー図

防御の視点では、〇〇情報がどの場所にあったとしても、攻撃者から READ されてはならないので、存在し得るすべての場所において「【攻撃者】は【〇〇情報】を【READ】できない」を保証する必要がある。そのため図 15-A-9-10 のように存在し得る場所ごとに子ノードを作成し、それらを論理積[^]で結ぶ。図 15-A-9-10 はノード 6 「【第三者】は【〇〇情報】を【READ】できない」に分解パターン【情報 READ 防御視点】を適用してノード分解した例である。

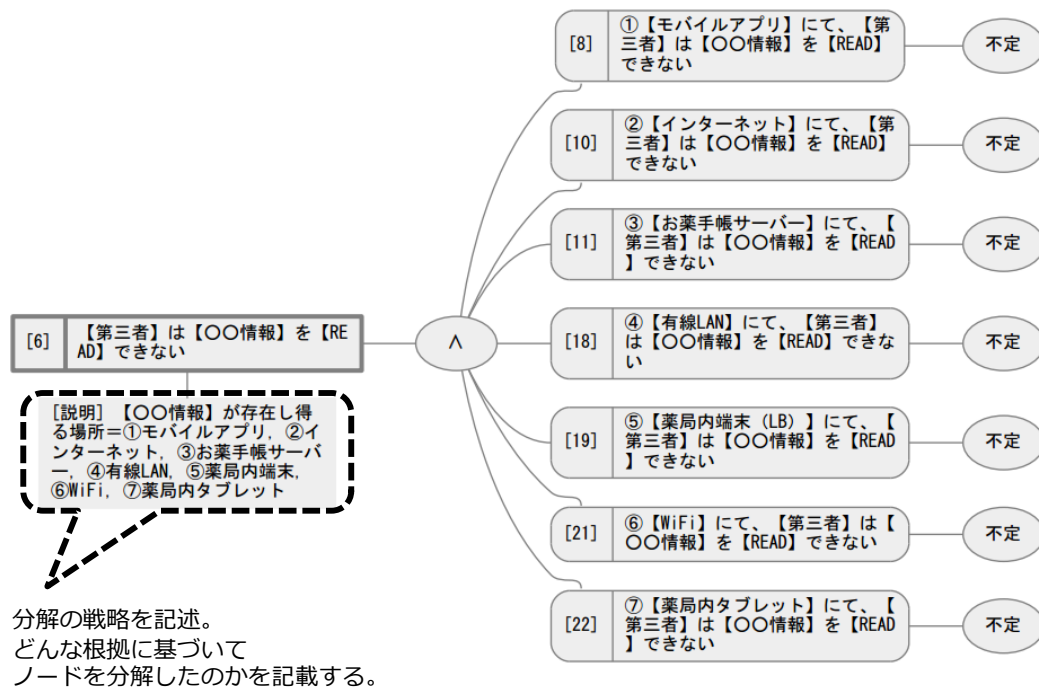


図 15-A-9-10 ノード分解例：【攻撃者】は【情報資産】を【READ】できない

図 15-A-9-10 では、情報資産が存在し得る「場所」に着目して、ノードを分解している。「場所」の概念は重要である。なぜなら情報資産が存在する場所に応じて、攻撃者の攻撃方法も異なるし防御する側の防御方法も異なるからだ。たとえばモバイルアプリ内に情報資産があるときはスマートフォンアプリに関する攻撃方法と防御設計を考慮する必要がある。インターネットを情報資産が伝達しているときには暗号通信に関する攻撃方法と防御設計を考慮する必要がある。それぞれ異なるシステムの設計情報やセキュリティの知識が必要となる。これが情報資産の存在する「場所」でノード分解する理由である。

場所に応じて分解したそれぞれの子ノードの TRUE、FALSE を判定するために、分析対象システムの設計情報をそれぞれ確認していくことになる。なお、情報資産の READ に関しては情報資産が伝達する方向については気にする必要はなく、どこに存在し得るかを洗い出すだけでよい。（これは、後述の WRITE には当てはまらない。）

4.2.2. 分解パターン【情報 READ 攻撃視点】

【攻撃者】は【情報資産】を【READ】できる
→ 【情報資産】が存在し得る場所でV展開する <ul style="list-style-type: none"> 一か所でも攻撃が成立すれば親ノードの命題が TRUE となるので論理和Vで子ノードを結ぶ

これは「誰それが〇〇情報を READ できる」と命題記述のあるノードの分解に使える分解

パターンである。4.2.1 の分解パターンとの違いは「できる」という語尾にある。攻撃者の視点において「できる」と表現しているのであるから、この命題が TRUE になるときは攻撃が成功している、つまり被害が発生しているという状況を表す命題となっている。

攻撃者にとっては、情報が存在し得るすべての場所のどこか一か所だけでも、情報を READ できてしまえば勝ちである。そのため図 15-A-9-11 のように存在し得る場所ごとに子ノードを作成し、それらを論理和 V で結ぶ。図 15-A-9-11 はノード 8 「【第三者】は【〇〇情報】を【READ】できる」に分解パターン【情報 READ 攻撃視点】を適用してノード分解した例である。

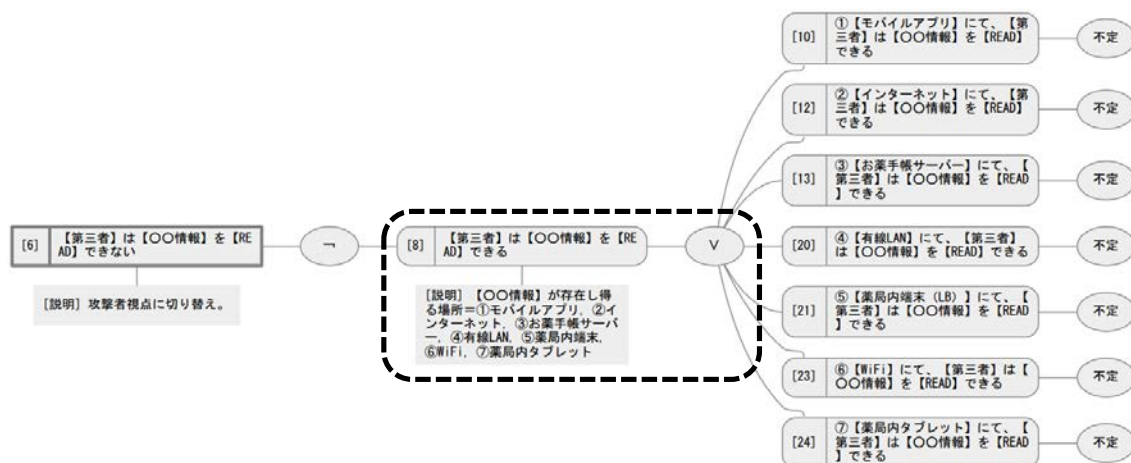


図 15-A-9-11 ノード分解例：【攻撃者】は【情報資産】を【READ】できる

なお、ノード 6 からノード 8 への否定「¬」を使った接続については 4.2.6 で説明する。

4.2.3. 分解パターン【情報 WRITE 防御視点】

【攻撃者】は【情報資産】を【WRITE】できない

- 【情報資産】の WRITE 結果が他のアクターにまで届くような WRITE 場所で
△展開する
- 攻撃者の影響が被害者に及ばない場所での WRITE は、誰も被害にあわないため、展開する子ノードには含めない
 - すべての防御が成立したとき親ノードの命題が TRUE となるので論理積「∧」で子ノードを結ぶ

これは「誰それが〇〇情報を WRITE できない」と命題記述のあるノードの分解に使える分解パターンである。分解手順は「【情報資産】の WRITE 結果が他のアクターにまで届くような WRITE 場所で△展開する」とある。少々分かりにくいだが次のようなことである。情報資産はシステム内のあちこちに伝達する。たとえば電子お薬手帳システムのとある情報資産

「□□情報」は図 15-A-9-12 のように①レセコン、②薬局内イントラ、③薬局内端末、④有線 LAN、⑤Wi-Fi 通信路、⑥薬局内タブレット、⑦インターネット、⑧お薬手帳サーバー、そして⑨モバイルアプリといった場所に伝達する。□□情報の伝達の方法は図 15-A-9-12 中に矢印で表現されている。⑦インターネットと⑨モバイルアプリの間の矢印は一方である。つまり□□情報は⑦インターネットから⑨モバイルアプリまで伝達するがその戻りはない。□□情報はモバイルアプリに表示されるだけの情報資産だからだ。

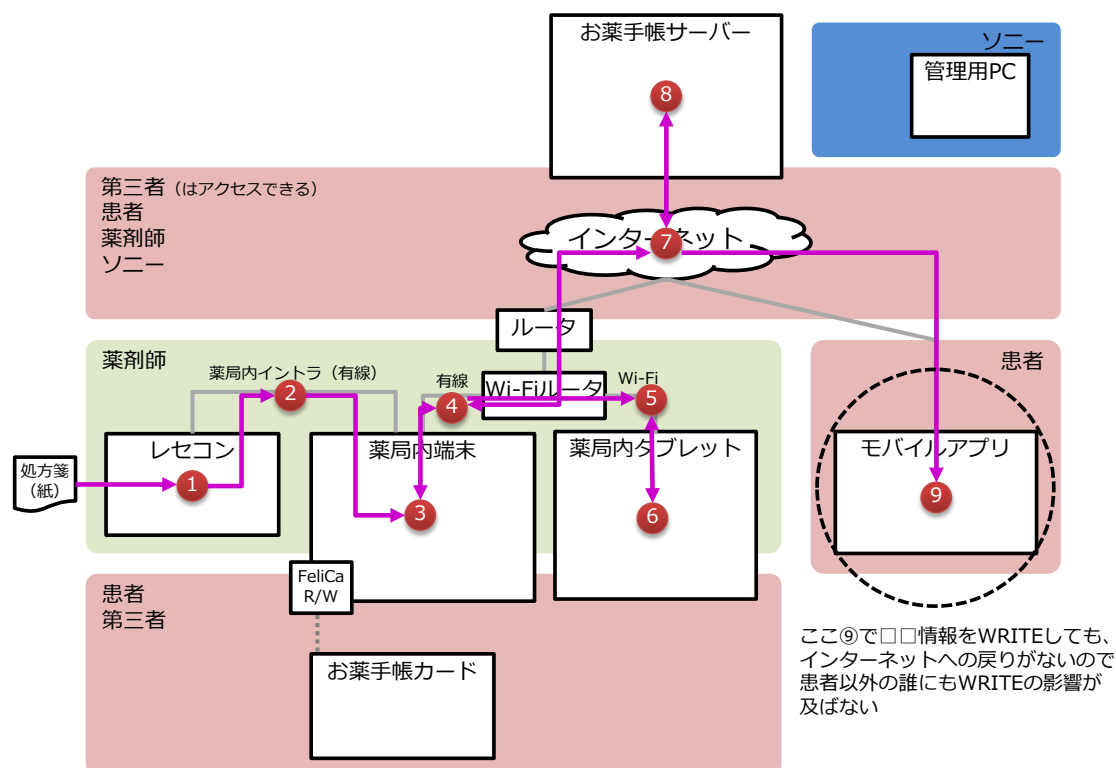


図 15-A-9-12 □□情報の伝達経路を洗い出したデータフロー図

ここでたとえばモバイルアプリのユーザーである患者が攻撃者となり、モバイルアプリを攻撃して□□情報の内容を **WRITE** できたとする。しかしその **WRITE** の影響はその患者自身にしか及ばず誰も困ることではない。たとえその患者自身が困ったとしても、それは自業自得であり、モバイルアプリのセキュリティ問題とはならない。よって分解手順「【情報資産】の **WRITE** 結果が他のアクターにまで届くような **WRITE** 場所で展開する」により作成する子ノードに⑨モバイルアプリは含める必要がない。

図 15-A-9-13 はノード 243 「【患者】は【自分の□□情報】を【**WRITE**】できない」に分解パターン【情報 **WRITE** 防御視点】を適用してノード分解した例である。展開された子ノードに⑨モバイルアプリが含まれていないことに注意してほしい。このように、情報資産の **WRITE** に関しては（つまり前述の **READ** と異なり）情報資産が伝達する方向についての注意が必要である。

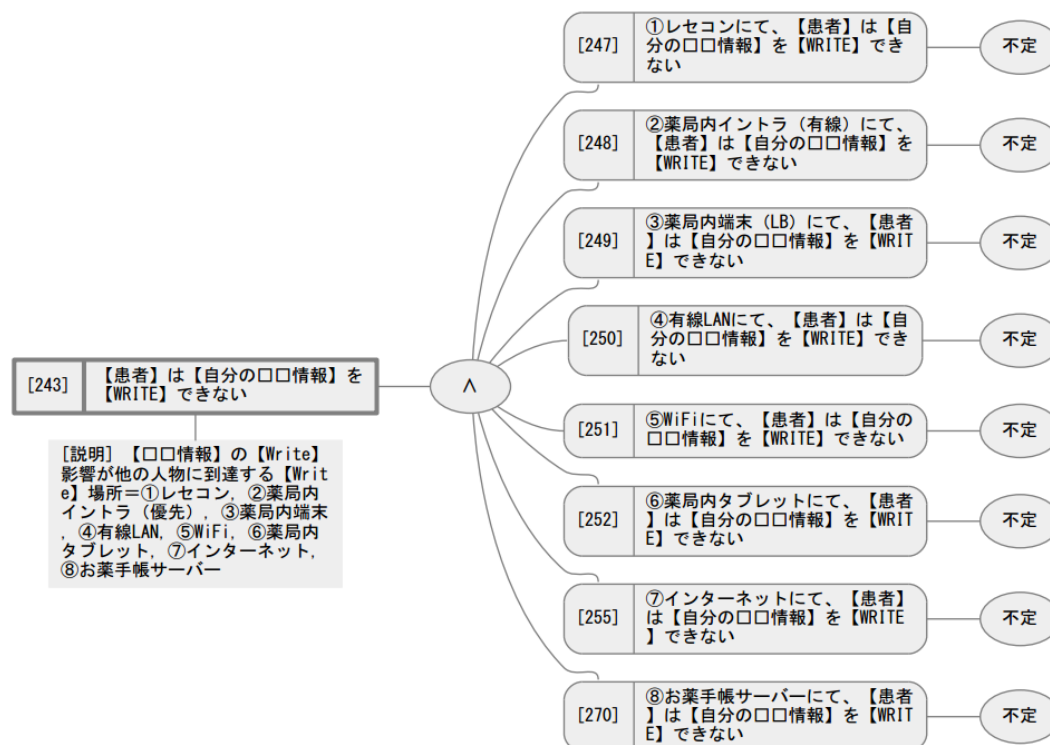


図 15-A-9-13 ノード分解例：【攻撃者】は【情報資産】を【WRITE】できない

4.2.4. 分解パターン【情報 WRITE 攻撃視点】

【攻撃者】は【情報資産】を【WRITE】できる

→ 【情報資産】の WRITE 結果が他のアクターにまで届くような WRITE 場所で V 展開する

- 攻撃者の影響が被害者に及ばない場所での WRITE は、誰も被害にあわないため、展開する子ノードには含めない
- 一か所でも攻撃が成立すれば親ノードの命題が TRUE となるので論理和 V で子ノードを結ぶ

これは「誰それが〇〇情報を WRITE できる」と命題記述のあるノードの分解に使える分解パターンである。4.2.3.の分解パターンとの違いは「できる」という語尾にある。攻撃者の視点において「できる」と表現しているのであるから、この命題が TRUE になるときは攻撃が成功している、つまり被害が発生しているという状況を表す命題となっている。

攻撃者にとっては一か所でも攻撃が成立すれば勝ちである。そのため図 15-A-9-14 のように子ノードを論理和 V で結ぶ。図 15-A-9-14 は、ノード 245 「【患者】は【自分の□□情報】

を【WRITE】できる」に分解パターン【情報 WRITE 攻撃視点】を適用してノード分解した例である。

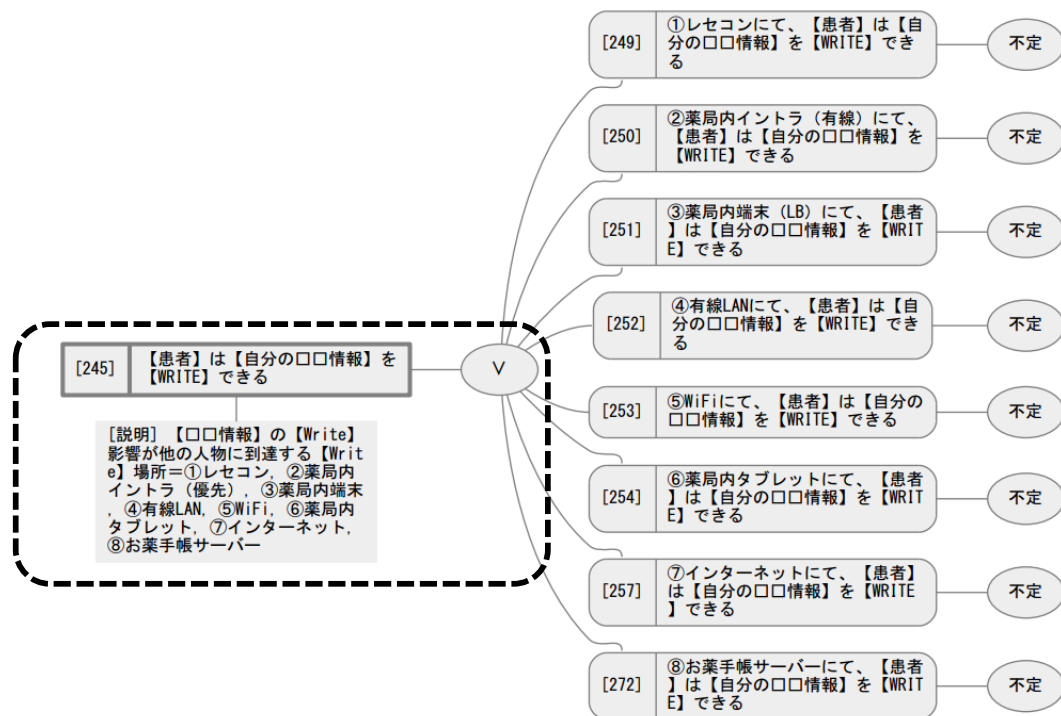


図 15-A-9-14 ノード分解例：【攻撃者】は【情報資産】を【WRITE】できる

4.2.5. 分解パターン【HTTPS 通信】

【情報資産】は【HTTPS 通信】で保護されている
→ 【HTTPS 通信】分解テンプレートにて展開する <ul style="list-style-type: none"> 分解テンプレートの子ノード

これは「【情報資産】は【HTTPS 通信】で保護されている」と命題記述のあるノードの分解に使える分解パターンである。図 15-A-9-15 は分解手順に記載のある【HTTPS 通信】分解テンプレートである。

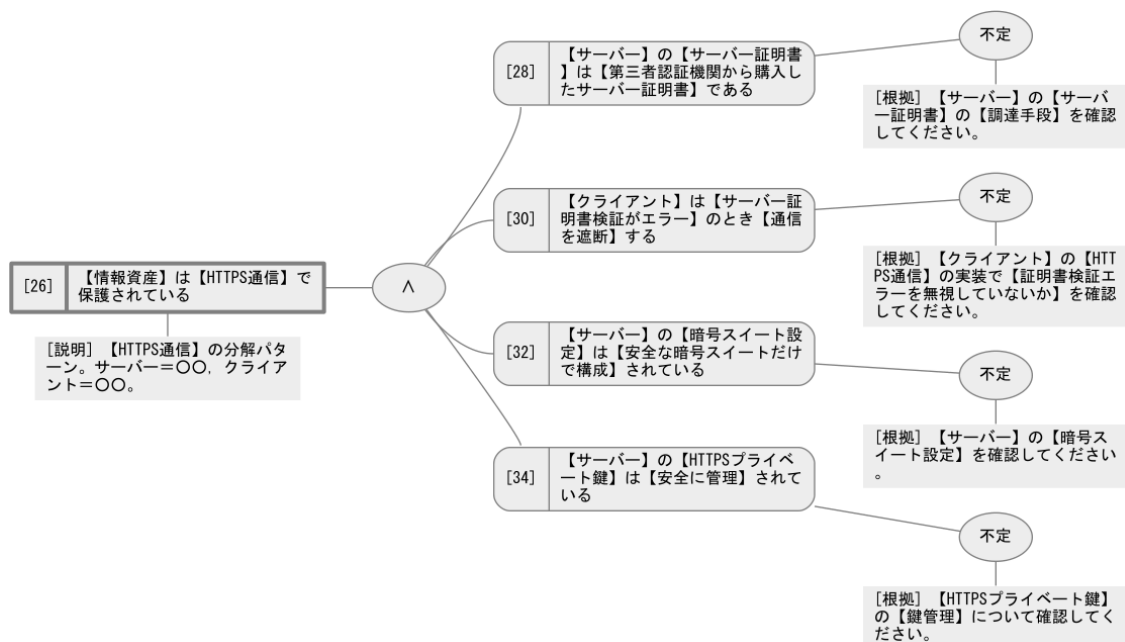


図 15-A-9-15 【HTTPS 通信】分解テンプレート

図 15-A-9-15 は「【情報資産】は【HTTPS 通信】で保護されている」という命題の親ノードが4つの子ノード群の論理積∧で表現できることを表している。これら4つの子ノード群は HTTPS 通信を正しく安全に利用するためのノウハウであり、これら4つのノウハウを正しく実施できているときに限り、「【情報資産】は【HTTPS 通信】で保護されている」と言えることを表している。

図 15-A-9-16 はノード 13「【薬局内端末とお薬手帳サーバーの通信内容】は【HTTPS 通信】で保護されている」に分解パターン【HTTPS 通信】を適用してノード分解した例である。

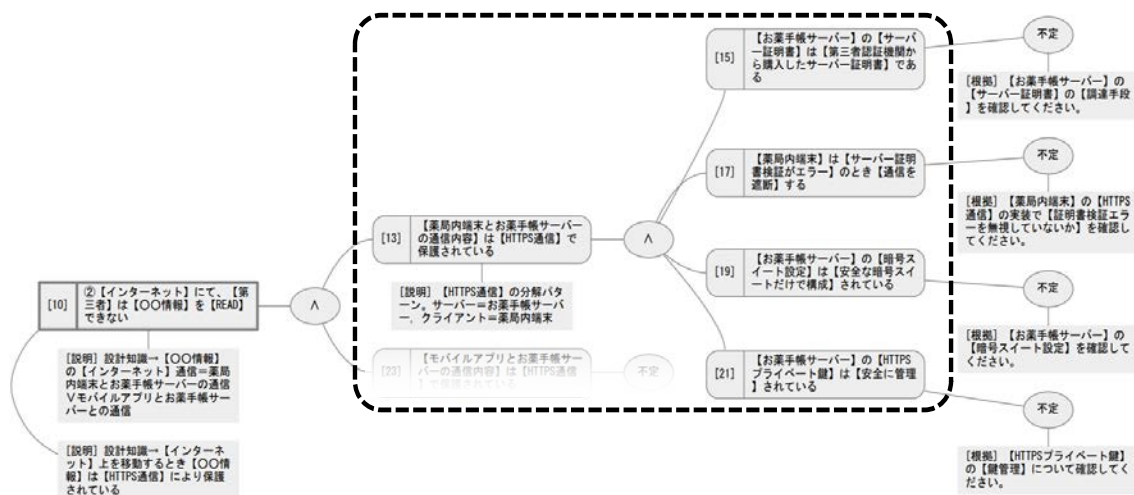


図 15-A-9-16 ノード分解例：【情報資産】は【HTTPS 通信】で保護されている

このように状況に応じて決まった子ノード群をコピー＆ペースト的に作成するだけでよいケースが多々あり、こうした状況ごとに分解テンプレートを知識ベース的に構築しておけば、分解テンプレートの再利用により分析木の構築を効率化できる。

4.2.6. 分解パターン【防御→攻撃切り替え】

【攻撃者】は【資産】を【操作】できない

→ 語尾「できない」を「できる」に変えて、否定一で接続

- ・ 防御視点のノード分解が難しく、攻撃視点のノード分解がしやすい場合に適用

防御視点のノード分解が難しく、逆に攻撃視点のノード分解がしやすい場合がある。たとえば HTTPS 通信のように確立した防御方法がないとき、攻撃方法を列挙するほうが簡単なことがある。そのようなとき、防御視点の命題「【攻撃者】は～できない」から攻撃視点の命題「【攻撃者】は～できる」に切り替えて、攻撃方法を子ノードとして展開するとよい。

図 15-A-9-17 はノード 6 「【第三者】は【〇〇情報】を【READ】できない」に分解パターン【防御→攻撃切り替え】を適用してノード分解した例である。

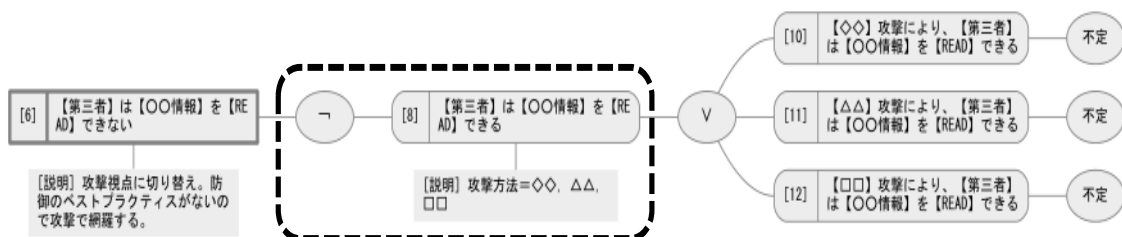


図 15-A-9-17 ノード分解例：【攻撃者】は【資産】を【操作】できない → できる

4.2.7. 分解パターン【攻撃→防御切り替え】

【攻撃者】は【資産】を【操作】できる

→ 語尾「できる」を「できない」に変えて、否定一で接続

- ・ 攻撃視点のノード分解が難しく、防御視点のノード分解がしやすい場合に適用

攻撃視点のノード分解が難しく、逆に防御視点のノード分解がしやすい場合がある。たとえば HTTPS 通信のように確立した防御方法があるとき、防御のための条件を列挙するほうが簡単なことがある。そのようなとき、攻撃視点の命題「【攻撃者】は～できる」から防御視

点の命題「【攻撃者】は～できない」に切り替えて、防御のための条件を子ノードとして展開するとよい。

図 15-A-9-18 はノード 15「【第三者】は【〇〇の通信内容】を【READ】できる」に分解パターン【攻撃→防御切り替え】を適用してノード分解した例である。

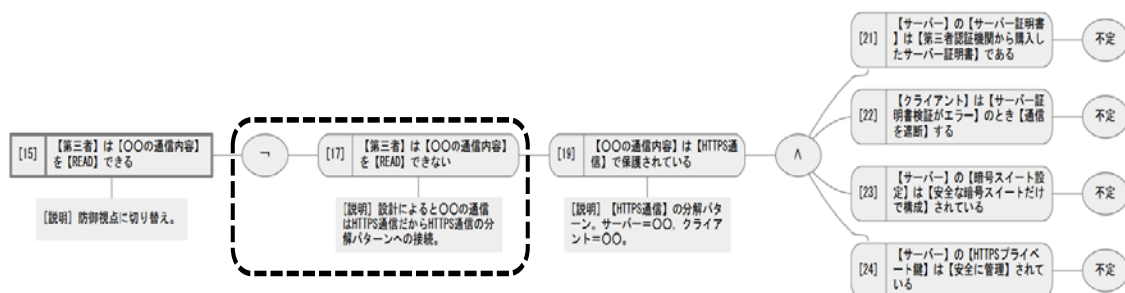


図 15-A-9-18 ノード分解例：【攻撃者】は【資産】を【操作】できる → できない

5. 効果

主観的な評価ではあるが、手順のパターン化によりセキュリティ分析作業で悩むことがなくなり、セキュリティ設計分析作業の効率化が図れた。複数人で分析するときにも分解パターンの活用により分析作業の効率化だけでなく、表現しようとしている意味の可読性も高まったと感じている。

6. 今後の発展に向けて

本編で紹介しきれなかった分解パターンもある。またセキュリティ設計分析業務を重ねるたびに新しい作業パターンや分解パターンができる。

セキュリティ設計分析に興味を持つ人々の間で、作業パターンや分解パターンを共有し充実させていくことができれば、セキュリティ設計分析の普及啓発に貢献できる。セキュリティ設計分析に興味を持ち研究してくれる人々が増えることを切に願う。

参考文献

- [1] Frank Swiderski, Window Snyder：脅威モデル セキュアなアプリケーション構築、日経 BP ソフトプレス、ISBN4-89100-457-6
- [2] Microsoft Developer Network：脅威モデルを作成する、
<http://msdn.microsoft.com/ja-jp/library/ff648644.aspx>
- [3] 松並 勝（Chief Security Technology Officer）：ソニーの電子お薬手帳システムに適用したセキュリティ設計分析手法、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、
http://www.prime-pco.com/wocs2015/program/B1510_AB-6.pdf
<http://www.ipa.go.jp/files/000043909.pdf>
- [4] 松並 勝：ねえねえやってる？仕様・設計のセキュリティ分析 ♪面白いよぉー、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、
http://isc2chapter.jp/wp-content/uploads/2014/03/仕様_設計のセキュリティ分析.pdf

掲載されている会社名・製品名などは、各社の登録商標または商標です。

独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター（IPA/SEC）