「ソフトウェア識別管理に向けた分析事業」の報告書

２０１６年3月９日

（独）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
セキュリティセンター
国際標準推進センター

目次

第１章　本取り組みの概要

1. 背景

近年の製品、システム、サービスは、ハードウェアだけでなく、ソフトウェアによって構成されるケースが増えてきている。これに伴い、ソフトウェア数量が膨大化するだけでなく、ソフトウェアのサプライチェーン（取引き関係や利用／運用／維持管理関係）も複雑化している。すると、色々な品質レベルのソフトウェアが世の中に溢れ、それを利用する際には信頼のおけるソフトウェアを見極めて使用することが重要となってくる。

最近の課題認識の事例としては、次のようなものがあげられる。

◆オープンソースソフトウェア（以下 OSS）である OpenSSL や struts において、バグ（脆弱性）が公表され、その対応について大騒ぎになった。製品やシステムを提供する事業者においては、それらのソフトウェアが自社の製品や　システムで使用されているかをいち早く把握し、対処を行ったり、利用者に注意喚起を行うことが必要になった。企業責任が問われる今日では、これは当然の反応であるが、当該ソフトウェアが自社で使用されているかどうかを素早く把握できた事業者は、そう多くはなかったものと推察できる。
それは、自社で使用しているソフトウェアを表面上の製品名では管理しているが、そのソフトウェアの内部で使用しているソフトウェアまではなかなか管理出来ていないと思われるからである。

◆今後のつながる世界（IoT 時代）においては、複数の異なる製品やサービスが接続されると想定される。ここで、IoT は Internet of Things のことなので、従来のMtoM通信（モノとモノの通信）のことを指すという解釈も可能だが、あらゆる組み込み製品やクラウドソフトウェアなどがつながる世界のことを指していると幅広く解釈願いたい。そこでは、接続相手のソフトウェアとつないでよいのかという判断が必要になる。そのためには、相手側ソフトウェアの各種情報が判断情報として必要となる。

2. 目的

上記の課題に対応するためには、ソフトウェア、及び当該ソフトウェアに関する各種属性情報（例えば、脆弱性情報等）を管理し、それをすばやく参照できる仕組みが必要である。これを、ソフトウェア属性情報管理と呼ぶことにする。しかしながら、このような管理には手間やコストがかかるため、民間の事業者での実施は難しく、限られた範囲での実施に留まっていると想定される。

そこで、ソフトウェア属性情報管理に関する将来の仕組み作りを念頭に置き、まずは実態調査や実現に向けての課題の抽出を目的として、この分析事業に着手することにした。

尚、各ソフトウェア開発会社等でツール等を用いて実施しているソフトウェアのバージョン管理等は、今回の議論対象ではない。

3. 実施内容

今回の活動は、次の３つからなる。

（１）　ソフトウェア属性情報管理の必要性、必要とする情報、管理の実態を事業者からヒアリング調査し、実現のための課題を抽出する。

（２）　既にソフトウェアの属性情報の一部を管理している事例が存在する。１つは資産管理用のデータベースであり、もう１つは脆弱性情報のデータベースである。
この２つを事例分析対象として、互いを突合し、それぞれの有用な情報の紐付けが出来ないかを検討する。

（３）　ソフトウェアを管理していく上では、それを識別する情報が必要となる。
これについては、全世界の共通的な指針が必要なため、いくつかの標準化団体で規格化に向けての活動が行われている。その状況を調査し、本活動全体の参考に資する。

第２章　ソフトウェア属性情報管理に関する検討

1. 背景

昨今のソフトウェアのサプライチェーンの変化に伴い、ソフトウェアの開発者や利用者から、次のような問題が挙げられるようになった。

◆OpenSSL や struts の脆弱性発覚などが示すように、自社製品やシステムに、安全性が問題になったソフトウェアが使われているかわからず、不安。

◆今後の IoT 時代において、相手のソフトウェアの情報を知らなければ、つないでよいか否かの判断ができない。

これら問題に対応するためには、“対象ソフトウェアに関する各種情報（脆弱性、ライセンス等）を管理しておき、参照できる仕組み”が必要になると考えられる。

2. テーマ

今回のプロジェクトでは、上記の仕組みの構築を将来目標とし、次項をテーマとして調査・検討を進めた。

（１）各事業者のソフトウェアの各種情報の管理の必要性、および重視するソフトウェア情報に関する認識調査

（２）各事業者におけるソフトウェア情報管理の実態・事例の調査

（３）ソフトウェア属性情報管理の実現に向けた課題の明確化、および解決の方向性についての検討

3. 進め方

前記のテーマ（１）および（２）については、いくつかの企業／団体を選んでヒアリングを実施し、実態を把握した。

前記のテーマ（３）については、文献調査およびヒアリング調査の結果を踏まえて検討を行った。

3.1 実態調査

3.1.1　調査対象の選定

対象企業／団体については、扱うソフトウェアのプログラム属性（市販パッケージ利用型、オープンソース利用型、スクラッチ開発型）と、扱われるソフトウェアの適用領域（組込、エンタープライズ、ネットワークサービス、ミッションクリティカル）の観点で、分類した。

これらのうちスクラッチ開発型については、スクラッチ部分のソフトウェアは自力で管理可能であり、スクラッチ以外のソフトウェアがある場合は、市販パッケージ利用型、オープンソース利用型に含められるので、今回の調査の対象外とした。（表 1）

表 1　調査対象の分類

	適用領域			
プログラム属性	組込	エンタープライズ	ネットワークサービス	ミッションクリティカル
市販パッケージ利用型	・F 社	・E 社 ・G 社	・C 社 ・D 社	・A 法人
オープンソース利用型	・F 社	・E 社 ・H 社	・B 社研究所 ・H 社	

3.1.2　ソフトウェア属性情報管理の必要性

調査前は、前述の表 1 のようにヒアリング対象を分類していたが、結果的には適用領域毎の差異はなかったため、プログラム属性に沿ってまとめた。（表 2）

表 2　プログラム属性毎の属性情報管理の必要性

プログラム属性	属性情報管理の必要性
市販パッケージ	・製品ベンダ、仕入れ業者の会社に対する信用で対応してきたが、パッケージソフトウェアの購入元から、意図しない（悪意の無い）使用に対してライセンス違反の指摘を受け、その賠償金を請求されるケースが増えてきており、ソフトウェア個々を管理しておく必要性が見直され始めた時期にある。
OSS	・OSS に含まれるソフトウェア群には、さまざまなライセンス上の制約があり、著作権を含むライセンス違反による、賠償金請求の発生などを避ける必要がある。 ・OSS に含まれる個々のソフトウェアの脆弱性が、OSS 全体の脆弱性をもたらす。そこを突いたサイバー攻撃の被害などを防止する必要がある。

以上により、市販パッケージおよび OSS のいずれにおいても、ソフトウェア属性情報管理が必要になってきているという認識である。

3.1.3　ソフトウェア属性情報として重視すべき情報

ソフトウェア属性情報管理に有用な属性情報としては、ソフトウェアを特定するための情報（製品名等）は当然として、その他には次の各項等が考えられる。

(1)ライセンス情報*1
(2)トレース情報*2
(3)脆弱性情報*3
(4)開発元情報
(5)品質情報
(6)使用実績
(7)市場の評価（風評）

上記のうち重要と考えられている情報は、ヒアリングによれば、ライセンス情報、トレース情報、脆弱性情報の３項目であった。
開発元、当該ソフトウェアの品質情報、実績、風評等もあり得るが、上記の３つに比べると、優先度は低いと捉えられていた。

……

*1 ライセンス情報：
開発者がそのソフトウェアの使用、改変、再配布、販売などの可否や条件を定めるライセンスを特定する情報である。
本稿では、そのソフトウェアの所有者、ライセンス発行者、ライセンス発行日、著作権所有者、著作権発生日等の情報とする。

*2 トレース情報：
ソフトウェア内部で更に別のソフトウェアを使用している際に、その使用関係を辿る情報である。

*3 脆弱性情報：
情報セキュリティ上の欠陥（可能性を含む）を示す情報で、本稿では IPA が JVN で脆弱性を指す次の各項目とする。
・概要：　脆弱性のエグゼクティブサマリ
・影響を受けるシステム：　影響を受けるシステムや製品、ライブラリなどの名称やそのバージョン番号など
・詳細情報：　この脆弱性に関する詳細な情報を記述する項目
・想定される影響：
脆弱性が悪用された場合、影響を受ける製品のユーザに対してどのような被害が想定されるか
・対策方法：　脆弱性を解消したりその影響を回避するための手段や手続き

……

3.1.4　各企業／団体におけるソフトウェア属性情報管理の実態

各企業／団体のソフトウェア属性情報管理の実態をまとめる（表 3）。

表 3　プログラム属性毎の管理状況

プログラム属性	現行の管理状況	課題
市販パッケージ	・大手企業では、資産管理、構成管理を実施しているが、中小企業では、部分的な資産管理のみに止まっている。 ・ライセンス管理は資産管理の範疇で、ライセンスの内容までは管理できていないケースが多い。 ・何か問題が発生した場合は、購入元の会社が対応してくれることを期待している。すなわち「会社の信用」で十分という考え方が根強い。 ・市販パッケージの品質等は、提供元が保証しているという前提である。それを利用しているシステムの品質は、中小企業、大手企業とも試験を通して確認している。	・自前での管理はコスト面から困難 ・業者とのパワーバランスで、問題発生時の対応の差に課題がある。
OSS	・大手企業の一部では、OSS 専門の組織をもち、自前でソフトウェアのソースプログラム自体を管理している。自前でソースプログラムを管理できない部分は、ラストリゾートの利用により対応可能としている。 一方、中小企業を含めた大半の企業では、ソースプログラムに踏み込んだ管理はできていない。 ・多くの種類のライセンス規約が存在し、利用上の制約も多岐にわたっており、一部の大手企業以外は十分な管理ができていない。 ・自由に利用できるため派生のソフトウェアが多く、脆弱性があるソフトウェアの混入の可能性が高いが、一部の大手企業以外は脆弱性情報と紐付けた管理はできていない。 ・脆弱性が発覚したソフトウェアが、コンポーネント[1]で使用されているか否かをすばやく判別するための管理は、一部の大手企業ではある民間企業が提供する OSS 照合サービスを利用している。それ以外の企業では、管理ができていない。 ・OSS を使用したシステムとしての品質は、中小企業、大手企業とも試験を通して確認している。	・自前での管理はコスト面、人材の確保が難しい。

1 コンポーネント…ソフトウェアプロダクトに含まれ、それ自体も実行可能な流通するソフトウェア。4.3.3 参照のこと。

3.1.5　ソフトウェア属性情報管理の事例

（1） データベースの例

(ア) 資産管理用データベース[2]

① 目的: 利用しているソフトウェアを判定し、ソフトウェア資産管理の構築と運用をサポートするためのソフトウェアデータベース
② 登録ソフトウェアプロダクト[3]：8 万件
③ 登録されている主なソフトウェア種別：市販パッケージ
④ ソフトウェア特定情報：インストール名称
⑤ データベース情報：ベンダ名、ソフトウェア名、バージョン、エディション、ソフトウェア別名、ソフトウェア種別（有償ソフトウェア、フリーウェア、ドライバ・ユーティリティ、HOTFIX、アドウェアなど）、ソフトウェア URL

(イ) オープンソースデータベース[4]

① 目的：、オープンソースソフトウェアに関するライセンス等の情報を含む包括的なデータベースで、セキュリティ脆弱性の監視および管理や、ライセンスのないコードや未承認のコードがソフトウェアプロダクトに侵入することの課題に対応
② 登録ソフトウェアプロダクト：50 万件
③ 登録されている主なソフトウェア種別：オープンソース
④ ソフトウェア特定情報：ソースファイル（ハッシュ値）
⑤ データベース情報：パッケージに含まれるファイル、ライセンス情報、著作権情報、脆弱性情報等
⑥ 利用イメージ：（図 1）

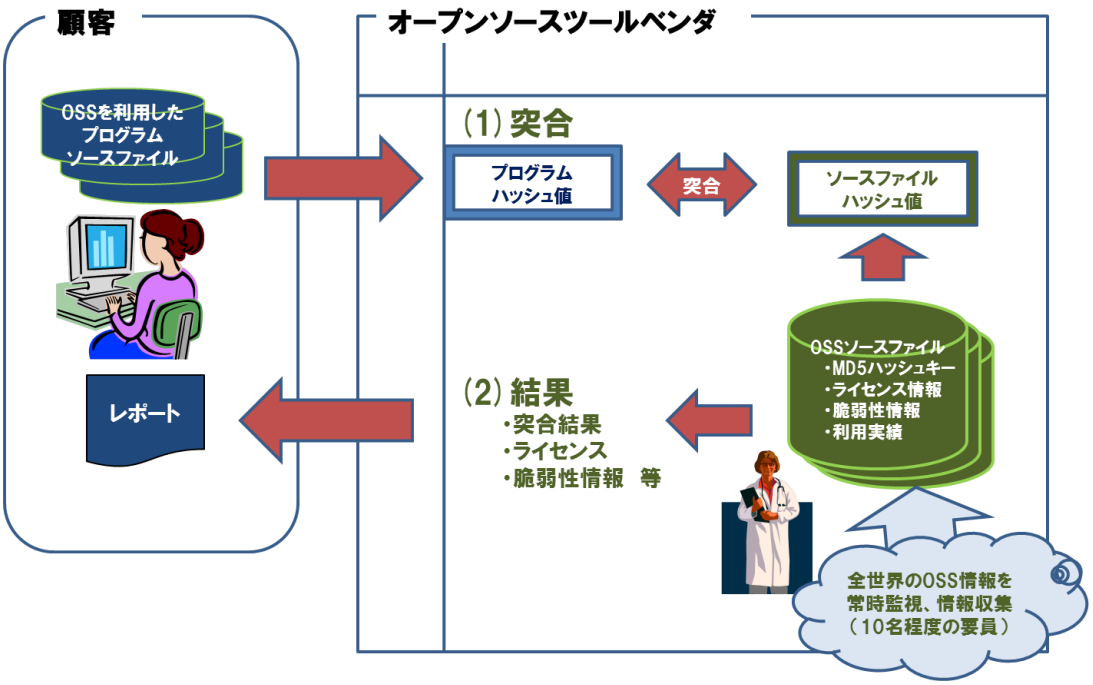


図 1　オープンソースデータベースによるソフトウェア属性情報管理の例

2 詳細は、別紙:ソフトウェアデータベース情報の「資産管理用データベース」の項を参照のこと。
3 ソフトウェアプロダクト…ソフトウェアの中で、開発者以外の人が利用できるように作られた、流通するひとかたまりのソフトウェア。4.3.1 も参照のこと。
4 詳細は、別紙:ソフトウェアデータベース情報の「オープンソース管理データベース」の項を参照のこと。

（2） 標準化もしくはそれに類する仕様

(ア) ISO19770-2[5]

① 目的：ソフトウェアの導入状況を把握するために、導入されたソフトウェアを識別するためのタグの規格

② 主体：国際標準化機構（ISO：International Organization for Standardization）と国際電気標準会議（IEC：InternationalElectrotechnical Commission）との合同技術委員会である JTC1（Joint Technical Committee 1 for InformationTechnology）配下の分科委員会 SC7（Subcommittee）下の作業部会である WG21（Working Group 21）

③ ソフトウェア特定情報：ソフトウェアタグ

④ データベース情報：ソフトウェア名称、ベンダ名、バージョン、当該ソフトウェアが含まれるパッケージ（オプション）

(イ) SPDX[6]

① 目的：ソフトウェアパッケージと関連コンテンツの情報の検出、収集、および共有を簡略化

② 主体：The Linux Foundation が支援する、ソフトウェア ベンダ、システム ベンダ、ツール ベンダ、各種財団、システムインテグレーターなど数十種の組織の代表

③ ソフトウェア特情報：ソースファイル

④ データベース情報：パッケージの正式名、バージョン、ファイル名、ダウンロード場所、ライセンス情報、著作権情報、脆弱性情報等

5 詳細は、別紙:ソフトウェアデータベース情報の「ISO-19770-2」の項を参照のこと。

6 詳細は、別紙:ソフトウェアデータベース情報の「SPDX2」の項を参照のこと。

4. 課題と解決の方向性

4.1検討項目の抽出

4.1.1　ソフトウェア属性情報管理で考慮すべき事項

ソフトウェアの属性情報を、データベースにより管理することをイメージする。
管理する情報としては、まずはニーズの高いライセンス情報、トレース情報、脆弱性情報の３つとし、他の情報は付加的に拡張管理可能なものとする。

このうち、ライセンス情報とトレース情報は、当該ソフトウェアの提供元が用意すべきものである。一方、脆弱性情報は、必ずしも提供元（開発元）が指摘するとは限らず、第三者によりもたらされる場合もある。
更に、前者２つは、ソフトウェアが開発された時に登録されるものであるが、脆弱性情報はそれが指摘された時点で登録されるものである。
もちろん、脆弱性情報を記載する欄は初期登録時にソフトウェア単位に用意されているのが望ましい。

4.1.2　二つのソフトウェア属性情報管理方式

以上を前提として、これらの情報を管理する方式として、次の２つが考えられる（図 2）。

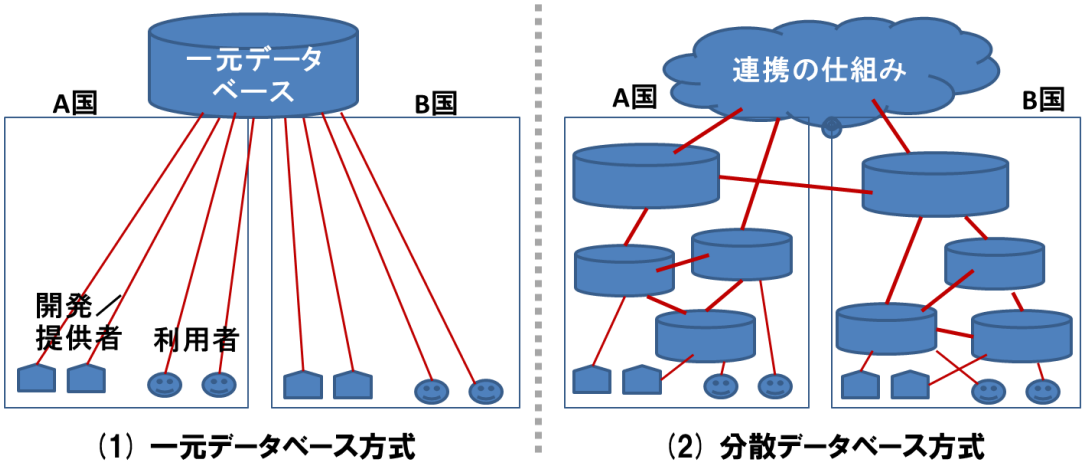


図 2 二つの属性情報管理方式

(1) 一元データベース方式
集中的に１箇所でソフトウェア属性情報を管理する方式。

(2) 分散データベース方式
例えば、各国のデータベースでそれぞれ属性情報を管理し、かつ、各国独自の組織単位に分散して管理するが、各データベースを連携させて、全体でソフトウェア属性情報を管理する方式。

OSS の場合は、これらを管理しようと考えた側が、オープンになった情報を自主的に集めてきて、データベースで一元的に管理する(1)の方式が可能である。

しかし、市販ソフトウェアの場合は、管理すべき情報が全てオープンになっているわけではないので、それらの情報を勝手に集めて一元的に管理するのは困難である。提供元が自主的に提供（登録）する仕組みが出来ることが必要となる。ところが、自由にデータベースへのソフトウェア登録を許すと、データベースへのサイバー攻撃（大容量／多数のソフトウェアを短時間に登録し、データ容量のオーバフローを引き起こしたり、不正なりすまし登録をすること等）が懸念される。従って、登録に際しては、全てを自動化することは危険であり、何らかの人の判断を介した正当な手続きの義務付けが必要である。すると、全体で１箇所の組織での登録受付事務は稼働的に不可能であり、(2)の分散管理とせざるをえないと考えられる。尚、登録受付事務の人手負荷のみを分散させればよく、性能的な条件さえ満たせば、データベースは必ずしも複数に分散されている必要はない。しかし、性能面や問題発生時の局所化を考慮すると、やはり分散データベース方式が妥当であろう。

登録時のルールや分散管理された情報を互いに参照可能とする方式は、今後の標準化方式として検討が進んでいく必要があると考える。

4.1.3　今回の検討項目

本稿では、分散データベース方式を想定したソフトウェア属性情報管理（特に、ライセンス情報、トレース情報、脆弱性情報）の実現に向け、以下の課題について、検討することとした。

（課題１）ソフトウェアの識別情報
（課題２）ソフトウェアの管理単位
（課題３）管理する情報の膨大化への対応
（課題４）管理するための手間とコスト
（課題５）各種情報の収集方法と管理方法

4.2 ソフトウェアの識別情報

4.2.1　検討項目

ソフトウェアの属性情報（特に、ライセンス情報、トレース情報、脆弱性情報）を管理可能とするためには、その前提としてソフトウェアを特定する情報が必要である。これを「ソフトウェアの識別情報」と呼ぶことにする。

その候補としては、ソフトウェア名称、製品名、識別子等、種々の情報が考えられるが、いずれが候補として妥当かを考える。

4.2.2　現状の識別情報の事例

（1）ある 資産管理用のデータベースでの事例

顧客システムにて使用されているソフトウェアを把握することが必要なため、各サーバ等に格納されているプログラムのインストール名称をキーとしてツールによる検索を行っている。この際の目的は、ライセンス等に関して注意すべきプログラムの抽出を簡易に行うことなので、インストール名称によるツールを用いた検索は目的に適うものとなっている。一方で、他の目的も意識した確実なソフトウェアの特定を行うには、インストール名称だけでは必ずしも当該プログラムのバージョン等まで特定できるとは限らないため、不十分な面もある。

（2） 脆弱性情報のデータベースでの事例

米国商務省国立標準技術研究所（NIST）が提供する脆弱性情報のデータベースでは、cpe（Common Platform Enumeration）[7] と呼ばれる情報により、ソフトウェアの検索を可能としている。一方で、cpe は各サーバのインストール情報を含まないため、これを用いて上述の資産管理の目的のための検索を行うには不適である。

以上のように、ソフトウェアの特定手段は、利用シーンに応じて異なるため、最初の検索キーとしてのソフトウェアの識別情報は個々に存在すると考えるのが妥当である。

ただし、その目的に応じてソフトウェアを特定した後は、関連する情報との紐付けやそれぞれの情報を持つ者同士の互いの共通理解のために、共通するソフトウェアの識別情報が存在することが望ましいと考えられる。

4.2.3　共通的な識別情報の検討

ソフトウェアの識別方法としては、次に示す２つの方法が考えられる。

（1）何らかのソフトウェアの識別子を付与し、それにより一意に特定可能とする。
（2）ソフトウェアのソースプログラム自身を特定情報とする。

（1）の事例としては、ISO/IEC19770-2 が標準化を進めているソフトウェアＩＤがある。

他には、ソフトウェアに限らず、あらゆるシステム資源に適用するＩＤとして、Ucode [8]がある。どの規格で統一するかは、世の中のコンセンサスにより形成されていくので、ここでは触れない。

（2）の事例としては、（3.1.5（1）（イ）で示した）米国の会社が提供しているサービスなどが存在する。

スタッフが世界の OSS を常に監視し、そのソースプログラム情報を取り込んだデータベースを管理している。そのサービスを受ける顧客は、自身の持つソフトウェアとハッシュ値で比較することにより、その一致性を確認することが出来る。

OSS の場合は、自由にソースプログラム情報を入手可能なことから、このようなデータベースの作成

7 情報システム、プラットフォーム、ソフトウェアパッケージを照合するため、これらに一意の名称を付与する仕様。Mitre 社で検討が進められており、2007 年 9 月に CPE 2.0 がリリースされている。

8 詳細は、別紙:ソフトウェアデータベース情報の「Ucode」の項を参照のこと。

が可能になっているが、市販のソフトウェアの場合は開発元企業からのソースプログラム情報の提供に抵抗が強いことが予想される。しかし、ソースプログラム情報を公開するのではなく、上記の比較のためだけの利用目的に限定し、データベースの中にハッシュ情報を持つだけならば、そのような抵抗感も薄まると想定される。実際、その取組みに賛同したコンソーシアムの中では、それを許容するソフトウェアベンダが拡大している。

4.2.4 識別情報の長所／短所

前述のように、共通的なソフトウェア識別情報としては、規格標準化された識別子を用いる方法とソフトウェアのソースプログラム情報そのものを用いる方法の２つが考えられる。この２つの方法の得失を考察すると、次のようにまとめられる（表 4）。

表 4　識別方式の比較

識別方式	得失	適用域
(1) 識別子を付与して管理	○誰でも統一的にソフトウェア同定が可能 ×識別子の付与や登録等の手間が開発元には負担	あらゆるソフトウェアを容易に特定する手段として有効
(2) ソースプログラム自体で管理	○開発元の負担が少なく、後からでも管理対象化可能 ×機械的仕組み（ハッシュ値比較等）でしかソフトウェア同定が難しく、人の理解では扱いが困難 ×市販品の場合は、ソースプログラムの提供に抵抗がある場合が想定される。	OSS のように、ソースプログラムを一元的に集められる場合に有効

両者の方法はともに有効ではあるが、得失もある。特に(2)はすでに民間サービスとして提供されているので、今後は識別子によってソフトウェア同定を可能とする(1)をベースとした方式の進展も期待される。

4.2.5 検討結果

（１）ソフトウェアを特定する目的は、その利用シーンによって異なるため、入り口の検索キーとしてのソフトウェアの識別情報を統一的に定めることは難しい。

（２）各利用シーンに応じてソフトウェアが特定された後、それとは別のソフトウェア情報と紐付けする際には、ソフトウェアの一意な識別情報が必要となる。
そのソフトウェア識別情報としては、規格標準となった識別子を用いるか、ソースプログラム情報を用いるかという２つの方法があり、適用領域に応じて使い分けられることが期待される。

4.3 ソフトウェアの管理単位

4.3.1　検討項目

ソフトウェアは、その中で別のソフトウェアを使用したり、あるいは別のソフトウェアとインタフェースを共有して連携して動作することがある。更に、ソフトウェアプロダクト（注１）では不具合対応時や機能追加時には、その修正の単位として、コンポーネント（注２）やモジュール（注３）といった単位が考えられる。

あらゆる目的を想定すれば、微細な単位のソフトウェアプロダクトで管理する必要が生じるが、手間やコスト、及び目的とするソフトウェアプロダクトを探すことの困難さを考慮すると、それは実情にそぐわない。重要視されているライセンス情報、トレース情報、脆弱性情報の３つを主に管理するための単位として何が適当かを検討する。

……

（注１）（注２）（注３）：ソフトウェアの構成単位について

本検討において、ソフトウェアの構成に関する単位については、開発言語の変化やプラットフォーム、開発ツールの進化で様々な呼び方が存在し、呼び方が同じでも違った範囲を指していることもある。
そこで本資料では、以下のように定義して説明することとする。

◎ソフトウェアプロダクト：
ソフトウェアの中で、開発者以外の人が利用できるように作られた、流通するひとかたまりのソフトウェア。

◎コンポーネント：
ソフトウェアプロダクトに含まれ、それ自体も実行可能な流通するソフトウェア。

◎モジュール：
ソフトウェアプロダクトに含まれ、それ単体では実行できないソフトウェア。

これらの相互関係を図示すると、図 3 のようになる。

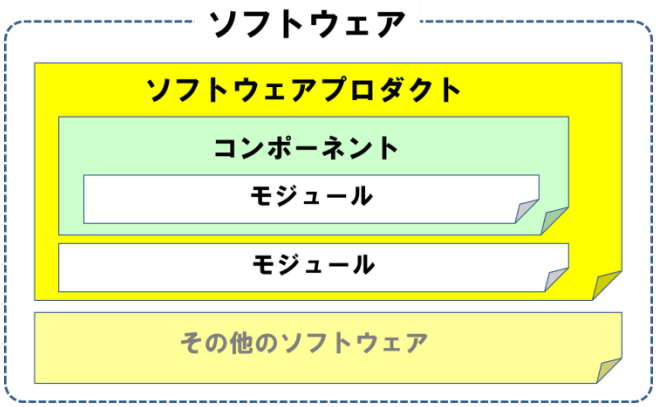


図 3　ソフトウェア構成単位の相互関係

4.3.2　現状の管理単位の事例

世の中のソフトウェアデータベースでの事例を、次に示す。

（1）ある資産管理用のデータベースの事例：
　資産管理用のデータベースでは、基本はインストールされたソフトウェアプロダクトとコンポーネントを管理している。
　これらに対するパッチ（HotFix：通常のリリース手順を踏まず緊急に発行される修正プログラム）も含まれているが、元のプロダクトやコンポーネントと紐付けて管理されているので、個別の管理対象とは捉えない。

（2）オープンソースソフトウェアデータベースの事例：
　コミュニティから提供されている OSS のソフトウェアプロダクト毎と、そのソフトウェアに含まれ呼び出される内部のコンポーネント単位で管理されている。

4.3.3　管理単位の検討

（1）ライセンス情報の管理に必要な単位と現状：
　第 3 者が開発したソフトウェアの利用にあたっては、そのソフトウェアのライセンス違反による賠償などの金銭的問題や違反行為の発覚による企業の信用低下という大きな影響がある。そのため、利用しているソフトウェアのライセンス管理は漏れのないきめ細かな対応が必要であり、ライセンスをもつソフトウェアすべてを管理する必要がある。
　通常、市販パッケージのライセンス情報はライセンス証書という形で、印刷紙面や製品ホームページなどで内容を確認できるが、その単位はソフトウェアプロダクトまたはコンポーネント単位である。
　OSS のライセンス情報は、対象のソフトウェアを入手できるコミュニティサイトから確認でき、またそのソフトウェアに含まれるコンポーネントに関しても、そのコンポーネントのコミュニティサイトで確認できるので、管理単位はソフトウェアプロダクトとコンポーネントと考えるのが妥当である。

（2）脆弱性情報の管理に必要な単位と現状：
　ソフトウェアの脆弱性情報は、直接的には個人情報の漏洩や金銭面の損失につながる場合があり、間接的にはセキュリティ事故による企業価値の低下につながるため、重要な情報である。しかし、市販パッケージにしても OSS にしても、直接的には脆弱性に関する情報を保持していない。これらの脆弱性に関する情報は、製品ベンダやコミュニティからの製品情報、利用者からのセキュリティ事故の情報、セキュリティに関する会社や団体などからのセキュリティ診断結果等、種々の情報源から得ることになり、その情報は色々な形式の情報となっている。
　それらの情報源は今後増減する可能性もあり、情報の形式の変化もあるので、統一的に考えることは難しい。そこで、IPA が管理している JVN iPedia を例に考える。
　IPA では、世の中で発生した脆弱性情報から、JVN iPedia の脆弱性情報のデータベースの更新と、発生したソフトウェアの情報を CPE Dictionary というデータベースに登録、紐付をしており、その紐付けの単位は、cpe 名という、ベンダ名、製品名、バージョン、アップデート、エディション、言語が含まれた識別子である。(CPE:Common Platform Enumeration)
　これらは、ソフトウェアプロダクトもしくはコンポーネントの単位に対応している。

（3）トレース情報の管理に必要な単位と現状
　今回の検討では、トレース情報とは、呼び出す側のソフトウェアと呼び出されて実行されるソフトウェアのライセンスが独立の場合に、両者を紐付ける呼び出し関係を示したものとしている。具体的には、呼び出し側のソフトウェアに記載される呼び出され側のソフトウェアに関する情報であって、ソフトウェアの所有者、ソフトウェア名、バージョン、エディションに関する情報としている。
これらの情報に該当するのは、市販パッケージ、OSS ともに、ソフトウェアプロダクトもしくはコンポーネント単位である。

4.3.4　検討結果

重要視されているライセンス情報、トレース情報、脆弱性情報の３つを管理するために必要となる管理の単位は、市販パッケージ、OSS ともに、ソフトウェアプロダクトもしくはコンポーネント単位であると考えるのが妥当である。尚、今回の検討では、流通しないソフトウェアを管理することには意味がない（作成元における管理で十分）と考えられるので、流通する単位とは考えにくいモジュールが管理対象外になっているのは妥当であると考えられる。

ただし、流通するソフトウェアの単位を考えると、ソフトウェアプロダクト単位が多いこと、及び脆弱性情報は大まかな範囲での注意喚起レベルの情報提供でも意味があることから、管理の容易性や維持管理コストの低減を考えると、実効的にはソフトウェアプロダクトのみの管理で十分な可能性もあり、今後も検討が必要と考える。

4.4 管理する情報の膨大化への対応

4.4.1　検討項目

ソフトウェアは年々増加傾向にあり、現在の自動車内に搭載されているソフトウェア量は、以前のスペースシャトルの６０機分にも相当すると言われている。これは、1つのソフトウェアの量が増えていることもあるが、ソフトウェアの数も増えていると想定される。

そのような世の中においては、ソフトウェアを管理しようとすると膨大な数のソフトウェアを対象にしなければならない。

ここでは、それを管理可能な有限の範囲内にとどめ、実現性を保つための方策について検討する。

4.4.2　現状の情報量の事例

（1）ある資産管理用データベースの場合

資産管理用データベースの管理元からの情報によれば、流通しているソフトウェアは 50 万~60 万件程度になる。しかし、ソフトウェアの詳細（メーカ名、バージョン、エディション等）がわからないものも多く存在し、インベントリー収集ツールで収集可能なものは、約 8 万件とのことである。

上記情報からできている資産管理用データベースでは、登録されている項目は sw_id、sw_vendor、sw_name、sw_sname、sw_version、sw_edition で、データはバリアブルである。そして登録件数は 8 万件で 12,099,584 バイト（12MB）となる。そのため、1 件当たりのデータ量を単純に計算すると、152 バイトである。

（2）OSS ツールベンダ保有のソフトウェアデータベースの場合

世界最大規模の OSS ナレッジベースは、5,000 以上のサイトから情報収集しており、登録ソフトウェア数が 50 万件以上である。そこには、下記の内容が含まれていると想定される。（内部仕様は非公開）

・パッケージの正式名
・パッケージのバージョン
・パッケージ ファイル名
・パッケージの説明
・ダウンロード場所　等

またソフトウェアに含まれるファイルのプロパティとしては、次のようなものがある。
・ファイル名（サブディレクトリも含む）
・ファイル タイプ（ソースプログラム、バイナリ、アーカイブ、その他）
・チェックサム
・ファイルに含まれているライセンス情報　等

また、ソースプログラムも含まれており、その総量は 1,000 億行ということである。プログラムの 1 行を 80 バイトと仮定すると総量は 8TB となる。プログラム 1 本当たり、パッケージの正式名などの他の情報は、1 件 1000 バイトとしても 50 万件で 0.5GB になる。合計で、データベースサイズは、多めに見たとしても 10TB 以下である。

4.4.3　管理する情報量の検討

資産管理用データベースの事例で 12MB、OSS ツールベンダのデータベースの事例では 10TB 以下である。仮に、それらでカバーできていないコンポーネントなどのソフトウェアプロダクトが存在しても、OSS の件数を超えることはないと考えると、多く見積もっても 20TB 以下と考えられる。

また資産管理用データベースでは、年間 6,000 件のソフトウェアプロダクトが追加になり、その影響は 1MB 程度である。OSS データベースの場合は年間で数万単位の増減があると言われているので、すべてが増加と考えたとしても、影響は 1TB 以下である。

一方、市場におけるハードディスク事情は、パソコンの周辺装置でも、1 台で 10TB をサポートできるハードディスクユニットが存在するため、初期 20TB、年間 1 TB 増加するとしても、初年度 3 台用意して数年単位で増設すればよい。

本稿では、分散管理方式によるソフトウェア属性情報管理を想定していることもあり、情報の肥大化に対して、記憶領域の確保はまだ深刻な課題にはならないと考えられる。

4.4.4　管理する対象と方式の検討

利用されなくなったソフトウェアプロダクトを、消し込む仕組みが必要になる。その方法には、次の２つが考えられる。それらの得失も併せて示す。

（１）　《頻度による消込み》
　一定の利用頻度しかないものを定期的に自動消去する仕組みを入れる。
（×利用頻度を監視する仕組みが必要）

（２）　《経年数による消込み》
登録から一定年数を経たもの（たとえば１０年）は、自動的に消去する。
（○経年ソフトウェアプロダクトは利用頻度が減り、メンテナンスも終了するので、実現が容易）

4.4.5　検討結果

今後のコンピュータや周辺装置の記憶容量の進化は、管理すべきソフトウェアプロダクトの純増には充分対応していけるものと考える。

したがって、情報の管理においては、情報量への配慮よりむしろデータの消し込み対応などの運用が重要と考える。

4.5 管理するための手間とコスト

4.5.1 検討項目

ソフトウェアの属性情報管理の必要性は、事業者には認知されつつも、現実に実施することにつながらないのは、手間とコストがかかるためである。この壁を打破するためには、最初の一歩として、必要となる手間やコストを明らかにすることが必要である。

それを踏まえて、その手間やコストの削減策の検討が実施され、徐々に現実のものとなっていくことが期待される。ここでは、その手間とコストを明らかにし、その後の削減策は今後の課題とする。

4.5.2 現状の管理稼働の事例

（1）ある資産管理用データベースの場合

資産管理のデータは年間 6,000 件の新規追加がある。これに要する稼働は不明。

（2）OSS ツールベンダ保有のソフトウェアデータベースの場合

年間で数万単位の増減があるといわれる OSS の更新を、当該 OSS ツールベンダでは、毎日 10 名以上の要員で対応しているとのことである。

4.5.3 想定される手間

ソフトウェアプロダクトの監視と管理データベースへの情報反映は、以下の情報が提供される時点が契機と考える。

（1）市販製品の場合

…市販製品の開発ベンダから情報が提供される時点

（情報の内訳）

- 新しく販売されるソフトウェアプロダクトとライセンス証書の提供情報
- 販売済みの製品に対する拡張製品やパッチ等のリリース情報
- 販売済みの製品名の変更情報[9]
- 販売済みの製品の販売終了情報[9]など

（2）OSS の場合

…オープンソース関連組織やその組織に参加する団体のサイトから情報が提供される時点

（情報の内訳）

- 新しく開設されるコミュニティと提供するプロジェクトの情報
- プロジェクトのソフトウェアが提供したライセンス情報
- プロジェクトにおけるバージョン管理情報
- コミュニティの活動状況
- コミュニティの活動停止の情報　など

上記の契機で、管理データベースの更新を実施する。

9 ベンダからこれら情報が提供されない場合もあるため、4.4.4 で示した経年数による消込みが必要になる。

4.5.4　考えられるコストの検討

（1） シミュレーションの仮定

ソフトウェアを管理する組織が我国に５組織あり、１つの組織で、年間に２千個程度※の登録事務を行うと仮定する。※：我国のソフトウェアプロダクトの開発数の妥当性は、今後検証の必要あり

（2） シミュレーション

・１人で１日に登録処理できる件数が３０個程度、１年を２００業務日程度と想定
　…10 人程度の組織体制が必要
・年間の必要経費は２億円程度（初期のシステム構築費等も考慮）
　…１件の登録あたり、10 万円程度の経費徴収が必要。

4.5.5　検討結果

ソフトウェア開発元企業にとって、登録するための上記負担額は微妙である。利用時のメリットは感じつつも、登録時のコストや手間を考えた際の総合判断は、意見が分かれるかもしれない。

ソフトウェア産業の信頼性向上のために、上記の必要性機運が高まることを期待したい。

4.6　各種情報の収集方法と管理方法

4.6.1　検討項目

ライセンス情報、トレース情報、脆弱性情報を一元的に把握できるデータベースを作成することを仮定して、それらの情報の収集方法や管理方法として意識すべき事項を検討する。

4.6.2　現状の各種情報の取得について

（1）ライセンス情報

市販パッケージの場合は、付属のライセンス証書、あるいはソフトウェアプロダクト説明書、ソフトウェアプロダクト紹介ホームページなどで情報入手できる場合が多い。

OSS の場合は、それを提供しているコミュニティのプロジェクトのサイトには、リンクするコンポーネントも含めたライセンス情報を記載している。また、3.1.5 項の図 1 で示したオープンソースデータベースの例では、ライセンス情報を調べて掲載している。

（2）トレース情報

市販パッケージの場合は、ソフトウェアプロダクト説明書等からその内部で利用している他のソフトウェアを把握できる場合もあるが、まだ一般的にはそのような習慣付けが行われているわけではなく、不明な場合が多々ある。

OSS の場合は、それを提供しているコミュニティのプロジェクトのサイトには、外部プロジェクト情報があり、リンクを追うことでコンポーネントのトレースができる場合が多い。そもそもソースプログラムを有する限りは、専門家ならばトレースを行うことは不可能ではない。しかし、専門家以外が行うのは難しいことや時間がかかることを考慮すると、陽にトレース情報が管理されていることが望ましい。

また、3.1.5 項の図 1 で示したオープンソースデータベースの例では、部分的なハッシュ値をとって突合できる仕組みがあるため、そのソフトウェアに含まれているコンポーネントを実質的に把握可能となっている。

（3）脆弱性情報

脆弱性情報は、そのソフトウェア開発時に存在するものではなく、世の中で広く使い始めてから第三者により指摘される場合が多い。従って、ライセンス情報やトレース情報と異なり、初期のソフトウェア開発時に開発元によって登録されるものではない。米国では、NIST が脆弱性情報データを有し提供している。我国でも IPA が NIST 情報をベースに脆弱性情報を提供している。

4.6.3　各種情報の収集方法の検討

（1）ライセンス情報

ライセンス情報は、そのソフトウェアの開発元が把握している情報である。従って、開発元からの申告情報が元となる。仮に、開発元からの申告が間違っていても、開発元の責任によるものなので、利用者はライセンス違反等を訴えられる危険は少ない。

しかし、第三者的なソフトウェア属性情報の管理組織がライセンス情報を入手しようとすると、現状ではライセンス証書が付いている当該市販パッケージを購入することが必要になる。民間でのあらゆるソフトウェアの購入は難しく、公的な機関が運営し、そこへの無償提供（むしろ登録には登録料が必要）のルールが必要と考えられる。あるいは、これに賛同する民間の有志団体に対して無償提供するルールでも構わない。

また、ライセンスを管理データベースに転記する際にミスを犯す危険もあり、その際に生じた被害を免責するようなルールも必要となる。

（2）トレース情報

トレース情報は、その開発元が管理している情報であるが、まずそれを開示する文化が根付いていない。また、その表記方法や開示の方法も確立されていない。更に、それが開示されたとしても、

過不足のない正しい情報であることが前提となる。意図的に誤った情報を開示することは考えにくいが、漏れや勘違いが発生する可能性は十分存在する。その際の責任を免責するようなルールを作らないと、情報提供元（開発元）の賛同を得にくいと想定される。

以上のように、トレース情報のイメージは感覚的には共有できるものの、第三者による管理とそこへの開発元からの情報提供を実際に実現しようとすると、表記方法や情報開示に関するルール作りを事細かに検討する必要がある。

（3）脆弱性情報

脆弱性情報の登録契機や登録者が、ライセンス情報やトレース情報と異なるため、最初のソフトウェア登録時には枠やポインタのみ作っておき、後で脆弱性情報が指摘された時点で、それに反映することになる。

ただし、脆弱性情報を監視する組織を新たに作ることは高コスト要因となるため、既に存在する脆弱性情報データベースにポインタを貼って管理するのが現実的であると考えられる。NIST や IPA のデータベースでは、ソフトウェアを特定する情報が cpe 名という、ベンダ名、製品名、バージョン、アップデート、エディション、言語が含まれた識別子である。これが統一的なソフトウェア識別子になるならば、互いを結びつけることは容易となるが、確実に一致することを確認するためには、統一番号や ID により特定できる方が望ましいと考えられる。

4.6.4 検討結果

ライセンス情報、トレース情報については、当該ソフトウェアの提供元による登録制にすれば実現が可能と想定される。ただし、あらゆるソフトウェアプロダクトの登録制実現には、何らかのルール作りやコスト負担の考え方の整理が必要である。

脆弱性情報については、既にセキュリティに関する情報を集めている組織（NIST 等）が存在するため、それと独立に脆弱性情報を収集・管理する仕組みを作る意義は薄い。ライセンスやトレース情報を含むソフトウェアプロダクト管理組織が出来た際に、脆弱性情報を管理する組織のデータベースとリンケージを貼って相互利用できる仕組みが出来ることが望ましい。それには、ソフトウェアの　　一致性を確認できる識別子の標準化が必要と考える。

5. まとめ

以上の結果を要約すると、次の通りである。

（1）ソフトウェア属性情報管理の必要性：
企業規模に関わらず、管理の必要性は感じている。

（2）管理が必要な情報の種類：
特に重要な情報は、ライセンス情報、トレース情報、脆弱性情報である。

（3） 管理の実態：
市販ソフトウェアについては、その提供元会社を信用した対応体制になっており、詳しい内部情報を管理していることはない。
一方、OSS に関しては、大企業は自前の OSS 組織を作ってソースそのものを管理したり、ラスト・リゾートと呼ばれる当該ソフトウェアに詳しい会社とサポート契約を結ぶことにより、問題が発生した場合の対応を行っている。しかし、中・小規模の企業は、コストの余裕がなく、自身が可能な範囲で調べて、問題なければ良しとしている状況である。

（4）ソフトウェア属性情報の実現に向けた見通し：
ライセンス情報、トレース情報については、当該ソフトウェアの提供元による登録制にすれば実現が可能と想定される。ただし、コストや手間がかかるので、その結果得られる安心感とのトレードオフでの判断となるので、あらゆるソフトウェアプロダクトの登録制実現には、何らかのルール作りやコスト負担の考え方の整理が必要である。今後の機運の高まりに期待したい。
脆弱性情報については、既にセキュリティに関する情報を集めている組織（NIST 等）が存在するため、それと独立に脆弱性情報を収集・管理する仕組みを作る意義は薄い。ライセンスやトレース情報を含むソフトウェアプロダクト管理組織が出来た際に、脆弱性情報を管理する組織のデータベースとリンケージを貼って相互利用できる仕組みが出来ることが望ましい。

今回の検討は、本稿の冒頭で述べた課題認識を端緒として、その議論のとっかかりを作ったに過ぎない。ソフトウェアの信頼性向上のために、いずれは必要になる議論であると考えており、今後も機会を捉えて取り組んでいきたい。

別紙：本文中で用いた略称組織名

Reference

SAMAC：一般社団法人ソフトウェア資産管理評価認定協会
IAC：The International Academy of Cytology
IOF：International Osteoporosis Foundation
FDA：Food and Drug Administration
SOUP：Simple Object Access Protocol
DEOS：Dependability Engineering for Open Systems
GPL：General Public License
NIPA：韓国情報通信産業振興院
DHS：United States Department of Homeland Security
NIST：National Institute of Standards and Technology
Linux　Foundation：The Linux Foundation
ISO：International Organization for Standardization
IEC：International Electrotechnical Commission
NICT：独立行政法人情報通信研究機構

別紙：ソフトウェアデータベース情報

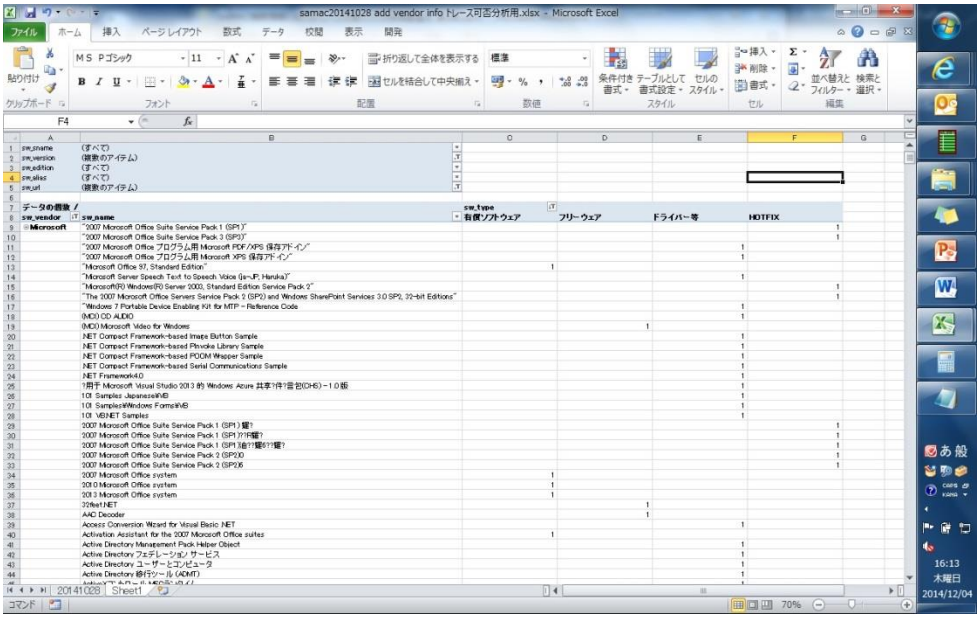
■　資産管理用データベース

ー　辞書に含まれる情報は下記の通り。

- 製品番号（sw_id）：SAMAC 内部番号
- ベンダ名（sw_vendor）：ソフトウェアのメーカ、著作権者、メーカ名、著作権者が不明の場合、販売会社を登録することもある
- 製品名（sw_name）：お客様から受領したソフトウェアのレジストリにあるインストール名称
- 略称（sw_sname）：sw_name のスペースを省いたもの
- バージョン（sw_version）
- エディション（sw_edition）
- 別名（sw_alias）：インストール名称をグループ化したもの
- ソフトウェア分類（sw_type）：SAMAC 内部で付与した分類（HOTFIX、アドウェア系、ドライバ・ユーティリティ等、フリーウェア、不明、文字化け、有償ソフトウェア）

ー　トレース可能と考えられるデータの選定

- データの内容と関連を調べるために、辞書の全カラムにデータが含まれているソフトウェアを選択（29391 件）したところ、sw_id、sw_vendor、sw_name、sw_sname、sw_version、sw_edition、sw_alias は当該ソフトウェア自体の情報であることから、sw_type、sw_url の情報に絞って調査。
- 選択したソフトウェアの sw_type は、有償ソフトウェア（8604 件）、フリーウェア（6743 件）、ドライバ・ユーティリティなど（12038 件）、HOTFIX（1857 件）、アドウェア系（149 件）だったが、トレースの可能性がありそうなソフトウェアは単独で実行可能なプログラムと考え、複数（少なくと 3 種類）の sw_type があるソフトウェアであるものを選択（234 社、14770 件）、さらに、調査期間も考慮して、上記選択ソフトウェアを多く保有するベンダの上位 10 社を選定



(Microsoft, National, Instruments, Hewlett-Packard, Oracle, FUJITSU, Intel, Autodesk, JUSTSYSTEMS, Adobe Systems, Canon) を選定（7786 件）、さらにトレース可能と考えられる製品の選定、ベンダ毎で類似する製品の内 sw_type が多いものを目視で選定した。

-

-

・ 選定した製品

- Microsoft：Exchange,SQL Server 関連
- National Instruments：LabVIEW 関連
- Hewlett-Packard：OpenView 関連
- Oracle：JAVA 関連
- FUJITSU：NetCOBOL 関連
- Intel：Visual Fortran 関連
- Autodesk：AutoCAD 関連
- JUSTSYSTEMS：一太郎関連
- Adobe Systems：Adobe Reader 関連
- Canon：imageWARE 関連

・ トレースや属性情報管理の可能性ある情報の選定

- 可能性のありそうな情報の候補は sw_url のみだったが、SAMAC の sw_url を参照した結果（参考資料参照のこと）から、記述されていた URL にある link 先まで追ったところ、動作環境、システム要件、必要なソフトウェアといった情報は探索可能だが、対象ソフトウェア自体に組み込まれているソフトウェア、ツール等の情報は探索できなかった。

■ オープンソース管理用データベース

ー BlackDuck のデータベースはコンサルビジネスの要であることから非公開であるため、BlackDuck 社からのヒアリングで確認できた類似する SPDX の情報をもとに、下記のような情報が格納されていると仮定している。

・ 対象のソフトウェアパッケージに関する情報

- パッケージの正式名
- パッケージのバージョン
- パッケージ ファイル名
- パッケージの説明
- ダウンロード場所
- 一意の識別子（ファイルと特定のパッケージとを紐付けるために作成されたもので、その値に影響を及ぼすことなく SPDX ファイルをパッケージに含めることができる）
- パッケージ内で宣言されているライセンス
- SPDX ファイルの作成者によって結論付けたライセンス
- パッケージ内のファイル レベルのライセンス リスト
- 著作権情報と日付

・ また、ソフトウェア パッケージに含まれるファイルのプロパティとして、次のようなものがあります。

- ファイル名（サブディレクトリも）
- ファイル タイプ（ソースプログラム、バイナリ、アーカイブ、その他）
- チェックサム
- ファイルに含まれているライセンス情報
- SPDX の作成者がファイルに適用すると結論付けたライセンス（たとえば、ファイル内にはライセンスがないが、同じディレクトリにコピー用ファイルがある場合など）
- 著作権所有者（掲載があれば）
- 著作権発生日（掲載があれば）
- ファイルの出自となる関連プロジェクト

・ トレースや属性情報管理の可能性ある情報の選定

- ライセンス情報、著作権情報、パッケージに含まれるファイル情報もあり、BlackDuck 社のヒアリングも含め、これらの情報をもとにトレースや属性情報は可能であり、更には NIST の脆弱性情報も BlackDuck のデータベースには保有していて、トレース結果から脆弱性のあるソフトウェアの有無を探索できる。また SPDX の V2 では、外部のコンポーネントのトレースも可能にするようである。

■ ISO-19770-2

- 国際標準規格としての ISO/IEC 19770
  ソフトウェア資産管理（SAM：Software Asset Management）に関する国際標準規格としては、国際標準化機構（ISO：International Organization for Standardization）と国際電気標準会議（IEC：International Electrotechnical Commission）との合同技術委員会である JTC1（Joint Technical Committee 1 for Information Technology）配下の分科委員会 SC7（Subcommittee）下の作業部会である WG21（Working Group 21）において策定が進められている ISO/IEC 19770 がある。現在、国際標準の規格として出版されているのは、プロセスの要求事項を規定した ISO/IEC 19770-1 とソフトウェアを識別するためのタグの仕様を規定した ISO/IEC 19770-2 である。今後この規格は、19770 シリーズとして SAM に関する規格やテクニカルレポート（TR：Technical Report）を発行する予定である。（2012 年 10 月 13 日現在）

- マイクロソフトは米国の規格団体や業界の主要グループと協力して、ソフトウェア識別タグの ISO/IEC 19770-2:2009 規格の開発を進めている。「InstallShield 2012 日本語版」のすべてのエディションで共通して「ISO/IEC 19770-2」のソフトウェア識別タグ作成機能が追加。

- Adobe は、2008 年の上半期に出荷された Acrobat9 で、業界で最初に ISO/IEC19770-2 ソフトウェア識別タグを実装した。2008 年の下半期に出荷された AdobeCreativeSuite4 にも、実装した。2009 年 5 月 13 日に ISO/IEC が提案した現在の国際標準最終案 v1 19770-2 にも対応している。

- ワールドワイドで多数の製品を販売しているベンダ中心の可能性が高いと思われる。

- SAMAC も参加している TagWG で議論されている ISO19770-2 にある「LINKS」情報は利用できないかを確認したところ、今後どうなっていくかは予想できないが、現在の最新バージョンの ISO19770-2 では「LINKS」はオプションフィールドであり、すでに出回っている製品はもちろん、今後世に出る製品に関しても「LINKS」を登録するベンダは限りなく少ないのではないかとのこと。また、ISO19770-2 の SWID はベンダ任せで（標準化団体の基本的な考え）、重複する可能性を SWID 登録時のベンダ Domain 名を付加して対応する、また、米国での非営利団体の TagVault やヨーロッパの団体で、ソフトウェアの認証と ID の払い出しをしているが、ISO19770-2 そのものを Microsoft や IBM であってもまだ一部の製品でしか使っていない。参考までに、確認した際の最新バージョンの ISO19770-2 は、以前のバージョンより必須登録情報が減ってきている。要因は、国防省の調達基準を満たす情報は必須とするのだが、市場に出る製品には Version,Edition 等で定義しきれない、例えばキャンペーン製品などに対しての対処ができないというベンダの強い反発からとのこと。

■ SPDX （Linux Foundation）

- 標準リストは 200 ライセンス近くで構成され、OSI（Open Source Initiative）が承認したすべてのライセンスも含まれている。SPDX ファイルは、タグ値（tag-value）フォーマット、または標準 RDF（Resource Description Framework）を使用して表すことができる。

- パッケージ、パッケージの内容、およびファイル レベル情報を識別するためのフォーマットを提供。説明対象のソフトウェアパッケージに関する情報には、以下のようなものがある。
  - パッケージの正式名
  - パッケージのバージョン
  - パッケージ ファイル名
  - パッケージの説明
  - ダウンロード場所
  - 一意の識別子（ファイルと特定のパッケージとを紐付けるために作成されたもので、その値に影響を及ぼすことなく SPDX ファイルをパッケージに含めることができる）
  - パッケージ内で宣言されているライセンス
  - SPDX ファイルの作成者によって結論付けたライセンス
  - パッケージ内のファイル レベルのライセンス リスト
  - 著作権情報と日付

- また、ソフトウェアパッケージに含まれるファイルのプロパティとしては、
  - ファイル名（サブディレクトリも）
  - ファイル タイプ（ソースプログラム、バイナリ、アーカイブ、その他）
  - チェックサム
  - ファイルに含まれているライセンス情報
  - SPDX の作成者がファイルに適用すると結論付けたライセンス（たとえば、ファイル内にはライセンスがないが、同じディレクトリにコピー用ファイルがある場合など･ 著作権所有者（掲載があれば)･ 著作権発生日(掲載があれば)
  - ファイルの出自（属性情報と考える）となる関連プロジェクト

- 一意にライセンスを識別でき、しかもファイル サイズをコンパクトにするために、この仕様書では、一般的なライセンス名のライセンス識別子一覧も提供しています（例えば、Apache 2 ライセンスならば “Apache-2.0”）

- NIST はデータソースパートナ(OSS のあらゆる情報を蓄積したナレッジベースと無料 OSS ディレクトリ Ohloh.net の完成度を高める情報提供元)
  The partners listed here are ones who actively support our automated data gathering efforts.

■ Ucode

- YRP ユビキタス・ネットワーキング研究所（以下、YRP UNL）が研究開発した、さまざまな「モノ」や「場所」などを識別するための国際標準の固有識別番号。
- オープンデータやビッグデータと IoT（Internet of Things：モノのインターネット）の技術基盤。
- ucode は 2012 年に ITU（国際電気通信連合）において、国際標準規格に採用され、IoT 向けの固有識別番号としては現在、世界で唯一の規格。
- 日本では火災報知機の個体管理に利用したり、競走馬の馬体にマイクロチップを埋め込み個体や血統管理をするなど、数千万個の利用例がある。また、欧州ではフィンランド、イタリア、フランスなどと共同実験を進めたり、アジアではタイと食品のトレーサビリティの試験事業を実施している。

■ SAMATE

- NIST SAMATE (Software Assurance Metrics And Tool Evaluation) プロジェクトは、ソフトウェアツールや技法の効果を評価・測定し、それらの差異を明らかにする技法を開発しソフトウェア品質保証を改善することを目的としている。

- ソフトウェア・ラベルについては、ユーザに対して情報を伝えるために、対象のソフトウェアはどういうものかを認識出来るよう、ソフトウェアのラベル（情報のテーブル）付けを行っていくという働きかけを始めている。プロジェクトでは、どういった情報を提供出来るのか、どういう情報を提供すれば本当に役に立つのか。そして、消費者、ユーザは何を必要としているのか等を検討している。

- SAMATE のプロジェクトにおいては、セキュリティ関連のプログラムを用意し、参加組織はツールを実行する。そして結果を我々にまた送り返してもらい、我々は送られてくる報告書から、どのツールがどういったものを見つけたか、ということをまとめる。ここでは、最良のツールを決めることが目的ではなく、どのツールがどのように改善出来るか等の情報を共有するのが意図である。静的解析ツールを使っていくことを推奨していくものです。現在は、SATE Ⅳ（追記※2014 時点でも最新と思われる）の段階ですが、既に一連のプログラムを用意しており、参加者に対してはこのプログラムを 7 月末に提供した。そしてこれから解析を行い、2012 年の 3 月 30 日にはワークショップを開いている。

- 属性情報管理の項目とかトレーサビリティのいう観点でなく、セキュリティを含むソフトウェアツールの効果をみてソフトウェアの品質保障を改善するプロジェクトである。IPA でも http://www.ipa.go.jp/security/awareness/vendor/programming/の IPA セキュア・プログラミング講座におけるツールプロジェクト情報（製品プログラマ向け）で、NIST/SAMATE（ Software Assurance Metrics And Tool Evaluation）が紹介されている。

第３章　JVN 脆弱性対策機械処理基盤での SWID タグ活用

1. はじめに

国内においても、JVN や JVN iPedia など脆弱性対策情報の提供環境は充実してきており、脆弱性の傾向などを把握しやすくなってきている。しかし、対策情報の多くは主に文書として構成されており、脆弱性の有無をチェックして対策を促すなど脆弱性対策に関わる処理の機械化については未だ発展途上にある。2009 年に流布した Web 誘導型マルウェアである Gumblar 以降、クライアントアプリケーションの脆弱性を悪用したマルウェア感染への対策は急務であり、クライアントアプリケーションを常に最新バージョンに維持することを促した。また、2014 年に報告された Apache Struts、OpenSSL（Heartbleed）、GNU Bash（Shellshock）などの OSS の脆弱性は、サーバにおいて影響を受けるコンポーネントを使用しているかどうか判断しにくいという新たな課題を投げかけた。

JVN で整備を進めている脆弱性対策に関わる処理の機械化を目指すフレームワーク MyJVN（JVN 脆弱性対策機械処理基盤）では、このような問題解決に向けて、脆弱性対策の処理の機械化や自動化を考慮した流通基盤の整備を進めてきている（図 6）。本章では、脆弱性対策に関わる処理の機械化の推進と共に、脆弱性対策の裾野を広げるために、製品識別子を用いた脆弱性対策と資産管理との連携について検討した結果を報告する。

2. JVN 脆弱性対策機械処理基盤

2.1 概要

JVN 脆弱性対策機械処理基盤は、図 4 に示す通り、JVN、JVN iPedia、MyJVN の 3 つのコンポーネントから構成されている。JVN（Japan Vulnerability Notes）は、情報セキュリティに関わるシステム管理者ならびにシステムエンジニア向けに脆弱性対策情報を広く告知することを目的とした情報公開サイトである。2003 年 2 月に JPCERT/CC の試行サイトとして運用を開始した。2004 年 7 月には経済産業省告示「ソフトウェア等脆弱性関連情報取扱基準」を受け、日本国内の製品開発者の脆弱性対応状況を公開するサイトとして情報発信を行なっている。2007 年 4 月からは、即時性と網羅性とを備えた情報提供を実現するため、JVN と JVN iPedia の 2 つのコンポーネント構成に拡張している。MyJVN は、処理の機械化や自動化を考慮した流通基盤であり、JVN と JVN iPedia に登録された脆弱性対策情報を用いたサービスを構築するフレームワークである。2008 年 10 月からフィルタリング型情報提供サービスを提供している。JVN と JVN iPedia の構成と外部連携を図 5 に示す。JVN iPedia では、国内の脆弱性対策の基点データベースとしての役割を果たすために、JVN ならびにアメリカ国立標準技術研究所 NIST が運用する NVD（National Vulnerablity Database）から脆弱性対策情報を取り込んでいる。また、JVN と JVN iPedia に登録されている脆弱性対策情報は、脆弱性対策情報のグローバルな情報源として NVD などの主要な脆弱性対策情報データベースから参照されている。

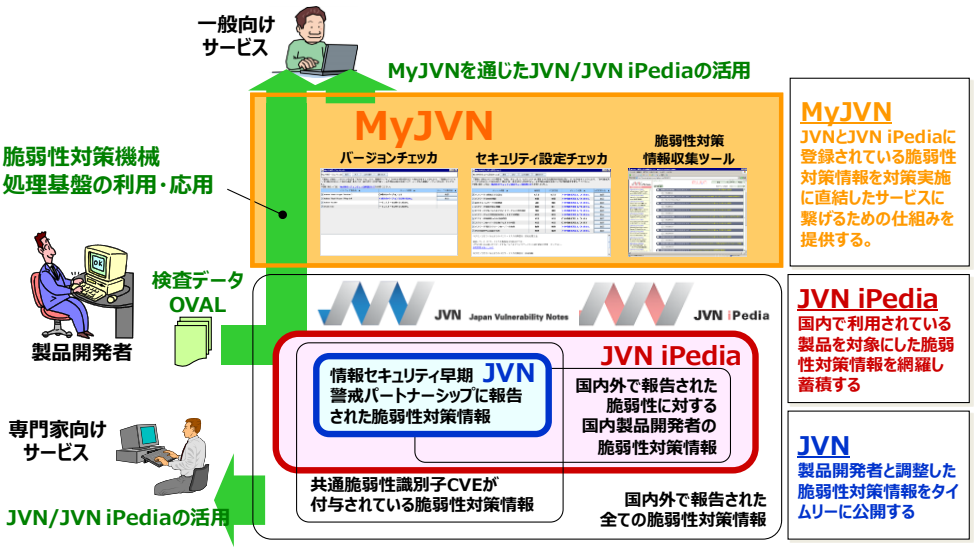


図 4　JVN 脆弱性対策機械処理基盤＝（JVN+JVN iPedia）×MyJVN

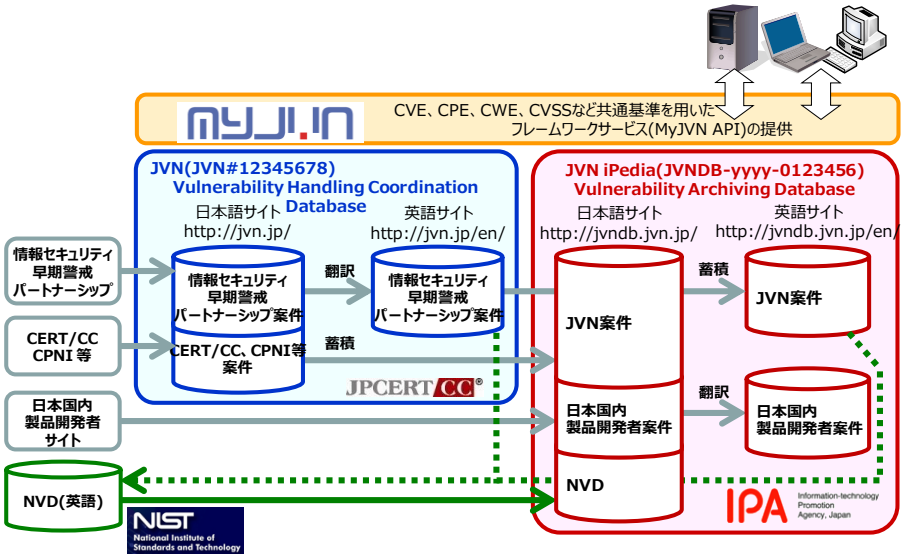


図 5　JVN と JVN iPedia の構成と外部連携

第1期 立上げ期

2002年 6月	JVNプロジェクトの開始
2003年 2月	JVN試行サイトの開設(慶応義塾大)
7月	JVNRSSフォーマットによる試行配信の開始
2004年 7月	情報セキュリティ早期警戒パートナーシップの開始
	脆弱性対策情報ポータルサイト JVNの開設
8月	脆弱性対策利活用(機械処理)基盤の検討開始
2005年 9月	JVNRSSフォーマットによる配信の開始
2006年 6月	JVNRSS試行サイトの開設(中央大)

第2期 対策情報充実期

2006年 9月	共通脆弱性評価システム(CVSS V1.0)多言語用計算機
2007年 3月	MyJVN API 試行サイトの開設(中央大)
4月	脆弱性対策情報データベース JVN iPediaの開設
	＋共通脆弱性識別子(CVE)
	＋共通脆弱性評価システム(CVSS)
2007年 6月	共通脆弱性評価システム(CVSS V2.0)多言語用計算機
2008年 5月	JVN 英語サイト、JVN iPedia 英語サイトの開設
9月	脆弱性対策情報データベース JVN iPediaのアップデート
	＋共通脆弱性タイプ一覧(CWE)
	CVE互換宣言(JVN iPedia)

第3期 利活用基盤整備・共通基準仕様導入期

2008年10月	脆弱性対策情報共有フレームワーク"MyJVN"の開始
	MyJVN脆弱性対策情報収集ツールのリリース
	＋共通プラットフォーム一覧(CPE)
2009年 4月	製品開発者の発信する脆弱性対策情報の自動収集の試行開始
6月	NIST NVD Official CPE Dictionary連携の試行開始
11月	MyJVNバージョンチェッカのリリース
	＋セキュリティ検査言語(OVAL)
12月	MyJVNセキュリティ設定チェッカのリリース
	＋セキュリティ設定チェックリスト記述形式(XCCDF)
	＋共通セキュリティ設定一覧(CCE)
2010年 1月	CVE互換取得(JVN、JVN iPedia、MyJVN)
	CVSS V2.0多言語用計算機(7ヶ国語)
2月	MyJVN API(フィルタリング型情報提供)の公開
2011年 3月	OVAL準拠認定取得(MyJVNバージョンチェッカ、MyJVNセキュリティ設定チェッカ)
	JVN iPedia/MyJVN情報のTwitter配信
8月	MyJVNバージョンチェッカのアップデート
	＋セキュリティ設定チェックリスト記述形式(XCCDF)
	＋評価結果形式(ARF)

第4期 利活用基盤整備・共通基準仕様普及期

2011年11月	サイバーセキュリティ注意喚起サービス"icat"の開始
2012年 8月	MyJVN API(検査データ提供)の公開
2013年 4月	ソフトウェア識別(SWID)タグ活用の検討開始
5月	OVAL準拠認定取得(MyJVN API)
2014年 9月	CVSS V2.0多言語用計算機(10ヶ国語)
11月	MyJVN脆弱性対策情報収集ツールV3のリリース
	＋共通プラットフォーム一覧(CPE)
	＋ソフトウェア識別子(SWID)
2015年 6月	CVSS V3.0多言語用計算機

図 6　JVN 脆弱性対策機械処理基盤の取り組み

2.2 製品識別子

2.2.1 共通プラットフォーム一覧（CPE）

MyJVN で採用している共通プラットフォーム一覧（CPE: Common Platform Enumuration）は、情報システムを構成するハードウェア、ソフトウェアの名称を、プログラムで（機械）処理しやすい形式で記述するための仕様である。また、CPE は、米国政府が推進している情報セキュリティにかかわる技術面での自動化と標準化を実現する技術仕様 SCAP（Security Content Automation Protocol）の構成要素のひとつである。米 MITRE によって開発され、2007 年にバージョン 1.0 が、2011 年にバージョン 2.3 が公開された。2012 年には、バージョン 2.3 が ITU-T：X.1528 として標準化されている状況にある。

記述形式は、cpe : /{種別} : {製品ベンダ名} : {製品名} : {バージョン} : {アップデート} : {エディション} : {言語}で、種別は、h=ハードウェア、o=OS、a=アプリケーションとなっている（図 7）。

cpe:/a:ipa:myjvn:1.0.0:update3:ed:ja

図 7　CPE バージョン 2.2 の例

なお、2008 年にリリースした MyJVN では、CPE バージョン 2.2 形式で『cpe : /{種別} : {製品ベンダ名} : {製品名}』までを記述した CPE データベースを実装している。

2.2.2 ソフトウェア識別（SWID）タグ

ソフトウェア識別（SWID: Software Identification）タグは、ISO/IEC 19770-2 で標準化された、ソフトウェアの導入状況を把握するために、導入されたソフトウェアを識別するための規格である。2009 年に初版が公開された。改訂版では、パッチ対応属性やハッシュ値属性のサポートなど脆弱性対策に関連する項目が盛り込まれ、2015 年 10 月に国際標準として発行された状況にある。しかし、該当製品の脆弱性に関わる情報との連携について充分とはなっておらず、利用面で環境整備が必要な状況にある。

記述形式は、ソフトウェア名、バージョン、製品ベンダ名などを XML フォーマットに表記する（図 8）。

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
  xmlns:swid="http://standards.iso.org/iso/19770/-2/2014-DIS/schema.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
  xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
  xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xsi:schemaLocation="http://standards.iso.org/iso/19770/-2/2014-DIS/schema.xsd schema.xsd"
  name="MyJVN API"
  version="1.0.0"
  tagId="ipa.go.jp+myjvn_api+1.0.0"
  versionScheme="multipartnumeric"
  >
  <swid:Entity
    name="独立行政法人 情報処理推進機構"
    regid="ipa.go.jp"
    role="softwareCreator"
    xml:lang="ja"/>
  <swid:Payload>
    <swid:File name="myjvn_api.jar"
      MD5:hash="583DE6974FCC12CCF6C7F189A8A51808"
      SHA1:hash="D1FA84BE3561E46A5C787DAD5FD1556F240B4863"
      SHA256:hash="3A91AC16AC90F354989314B65F2C73E0365D80087760B7D833D8F6BD7813F01F"/>
  </swid:Payload>
</swid:SoftwareIdentity>
```

図 8　SWID タグの例

2.2.3 CPE と SWID タグとの関連付け

CPE と SWID タグとの関連付けについては、SWID タグの導入、相互運用可能な SWID タグの作成のガイドラインである NIST IR 8060 において検討されている。この Draft3 のガイドラインの中で、図 9 に示すように、SWID タグから CPE バージョン 2.3 を生成する手順が示されている。

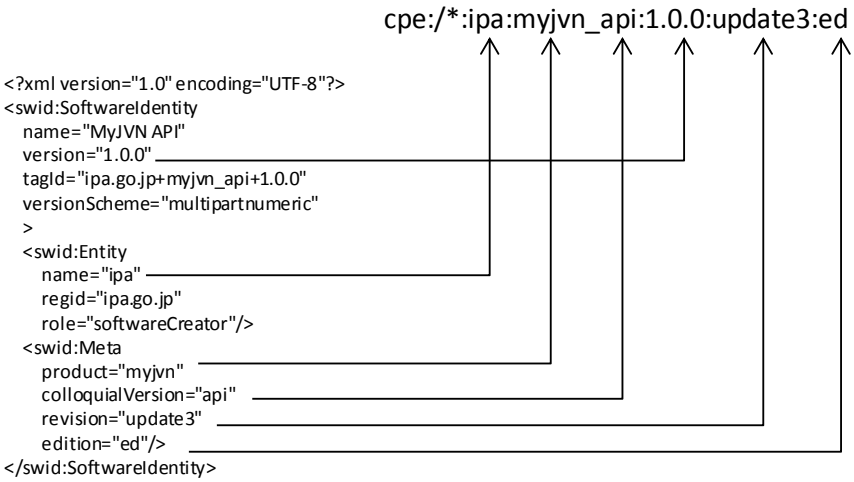


図 9　SWID タグから CPE バージョン 2.3 の生成手順

なお、CPE バージョン 2.3 とバージョン 2.2 は、フォーマットの相互変換が可能な仕様になっている（図 10）。このため、以降の説明では、CPE 製品名/CPE 名としてバージョン 2.2 を前提とする。

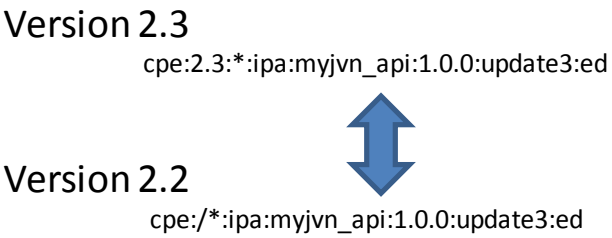


図 10　CPE バージョン 2.3 と 2.2 のフォーマット

2.3 JVN 脆弱性対策機械処理基盤整備にあたっての課題

本分析事業では、JVN 脆弱性対策機械処理基盤整備にあたり、次の 2 つの視点から検討を行った。

- 脆弱性対策をより一層推進するためには、どうしたら良いのか？
- 国内において、脆弱性対策の対象となりえる製品の件数規模は、どのくらいあるのだろうか？

一つ目の課題は、脆弱性対策の裾野を広げるためのアプローチについての課題である。この課題については、JVN 脆弱性対策機械処理基盤の普及による利便性の向上、資産管理/インベントリ管理などデータ連携などによる多様な脆弱性対策の推進により解決できるものと考えている。このため、本分析事業では、脆弱性対策の裾野を広げるために、製品識別子を用いて資産管理/インベントリ管理と脆弱性対策との連携の可能性を探ることとした。

二つ目の課題は、資産管理/インベントリ管理と脆弱性対策との連携で使用する製品識別子の件数規模の見積もりについての課題である。一般的に、JVN iPedia、NVD などの脆弱性対策情報データベースに登録されている製品は、脆弱性が報告されて初めて該当製品としてデータベースに登録される。一方、資産管理/インベントリ管理では、脆弱性が報告されていない製品も管理対象となっている。このため、双方に登録されている製品と、どちらか一方にしか登録されていない製品がでてくることになる。本分析事業では、片方にしか登録されていない製品情報を、どのようなデータとして持ち合うべきなのかを考えるにあたり、その規模を探ることとした。

3. SAMAC ソフトウェア辞書との連携の可能性

本節では、一般社団法人ソフトウェア資産管理評価認定協会（SAMAC: association of SAM Assessment & Certification）が保有している SAMAC ソフトウェア辞書とのデータ連携について報告する。SAMAC は、ISO/IEC19770 もしくはそれに関連する JIS 規格に準じた評価基準を開発、利用を行う組織である。SAMAC ソフトウェア辞書を開発し、協会会員に提供している。

3.1 目的

2014 年に報告された Apache Struts、OpenSSL（Heartbleed）、GNU Bash（Shellshock）などへの対処を通して、脆弱性対策には、資産管理やインベントリ管理が必要であることが再認識された。脆弱性対策をより一層推進するためには、「JVN 脆弱性対策機械処理基盤の普及」、すなわち、資産管理/インベントリ管理とのデータ連携が極めて重要である。

現在、多くの場合、ソフトウェアのインストール状況と脆弱性との紐付けは人手で行われている（図 11）。すなわち、新たに報告された脆弱性の影響を受けるソフトウェアの有無、影響を受けるシステムの特定、影響を受けるホストの特定のために、資産管理/インベントリ管理と脆弱性管理が連携するに至っていない。

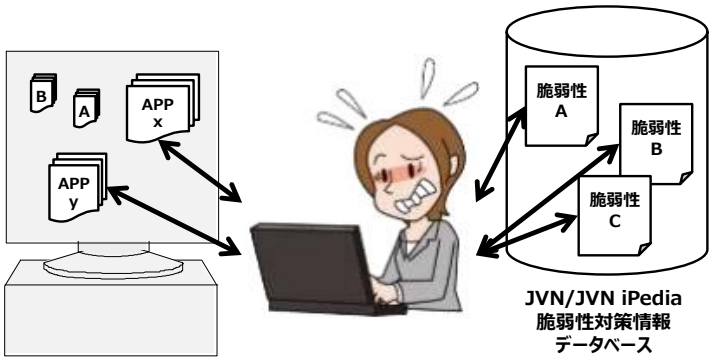


図 11　人手によるインストール状況と脆弱性との紐付け

SAMAC ソフトウェア辞書と JVN/JVN iPedia 脆弱性対策情報データベースとの紐付けができると（図 12 の橙色の線）、資産管理/インベントリ管理を利用してインストール状況と脆弱性との紐付けができることになる（図 12 の青色の線）。すなわち、インストール状況に合わせた脆弱性対策の推進につながるだけではなく、資産管理/インベントリ管理ツールから、脆弱性管理への歩み寄りが容易になる。

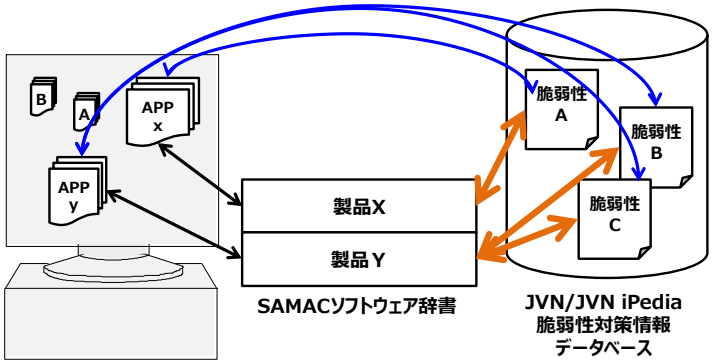


図 12　データ連携によるインストール状況と脆弱性との紐付け

そこで、SAMAC ソフトウェア辞書と JVN 製品データベースとの紐付けを通して、次の課題について検討する。

(a)データ連携の実現方法と、連携のための実現上の課題を明らかにすること
(b)今後、JVN 製品データベースに登録される可能性のある製品件数を把握すること
(c)JVN 脆弱性対策機械処理基盤の製品データベースの改善課題を明らかにすること

3.2 SAMAC ソフトウェア辞書

SAMAC ソフトウェア辞書に登録されている項目は、製品ベンダ名、ソフトウェア名、エディション、バージョン、ソフトウェア種別（有償ソフトウェア・フリーウェア、HOTFIX、ドライバ・ユーティリティ等）となっている（図 13）。2014 年 10 月末に、SAMAC から入手した SAMAC ソフトウェア辞書の登録項目は、表 5 の通りである。

ソフトウェア名	ベンダ名	エイリアス	バージョン	エディション	種別
Adobe Flash Player 10 ActiveX	ADOBE SYSTEMS	Flash Player	10	ActiveX	フリーウェア
Realtek High Definition Audio Driver	Realtek Semiconductor	High Definition Audio Driver	-	-	ドライバー・ユーティリティー等
Microsoft .NET Framework 3.5 SP1	Microsoft	.NET Framework	3	-	フリーウェア
IP Messenger for Win32	白水 啓章	IP Messenger	32	-	フリーウェア
Microsoft Office Personal 2007	Microsoft	Office	2007	Personal	有償ソフトウェア
JUSTSYSTEMｱﾌﾟﾘｹｰｼｮﾝの追加と削除	JUSTSYSTEMS	ｱﾌﾟﾘｹｰｼｮﾝの追加と削除	-	-	ドライバー・ユーティリティー等
Google Toolbar for Internet Explorer	Google	Google Toolbar	-	-	フリーウェア
Intel(R) Graphics Media Accelerator Driver	Intel	Graphics Media Accelerator Driver	-	-	ドライバー・ユーティリティー等

図 13　SAMAC ソフトウェア辞書

表 5　SAMAC ソフトウェア辞書の登録項目

#	項目	概要
1	sw_id	SAMAC 辞書用のソフトウェア識別番号で、内部 ID となっている。ソフトウェア識別番号として、96407 までが付与されている（2014 年 10 月 28 日時点）。
2	sw_vendor	製品ベンダ名で、不明の場合には、”-“が記載されている。
3	sw_name	ソフトウェア名
4	sw_sname	ソフトウェア名短縮名であり、ソフトウェア名から空白が削除されている。
5	sw_version	バージョン
6	sw_edition	エディション
7	sw_alias	別名
8	sw_type	ソフトウェア種別であり、下記が記載されている。 ● 有償ソフトウェア ● フリーウェア ● HOTFIX ● ドライバ・ユーティリティ等 ● アドウェア系 ● 不明 ● 文字化け
9	sw_url	製品の URL

なお、この SAMAC ソフトウェア辞書は、インベントリ収集ツールで収集可能な[プログラムの追加と削除]に表示されているインストール名称をベースに作成された一覧となっている。このため、同じ製品、バージョンであっても、ソフトウェア名が異なることがある。図 14 に JRE 1.4.2_03 の事例を示す。図 14 は、いずれも同じ製品バージョンであるが、上段はソフトウェア名に空白なし、下段はソフトウェア名に空白ありとなっていることから、SAMAC ソフトウェア辞書上では、これら 2 つの製品は、異なるものとして辞書登録されている。

Java 2 Runtime Enviroment,SE v1.4.2_03
Java 2 Runtime Environment, SE v1.4.2_03

図 14　ソフトウェア名が異なる事例

（1）辞書エントリ数
SAMAC ソフトウェア辞書に登録されている辞書エントリ数は、次の通りである（表 6）。

表 6　SAMAC ソフトウェア辞書に登録されている辞書エントリ数

項目	件数
辞書エントリ数	86,103 件

（2）製品ベンダ数
SAMAC ソフトウェア辞書に登録されている製品ベンダ数は、次の通りである（表 7）。

表 7　SAMAC ソフトウェア辞書に登録されている製品ベンダ数

項目	件数

製品ベンダ不明の辞書エントリ数		11,440 件
製品ベンダ記載の辞書エントリ数		74,663 件
	製品ベンダ数	一意の製品ベンダ名を機械的に識別すると、製品ベンダ数は 7,688 件

(3) ソフトウェア種別

86,103 件のソフトウェア種別の内訳は、次の通りである（図 15）。

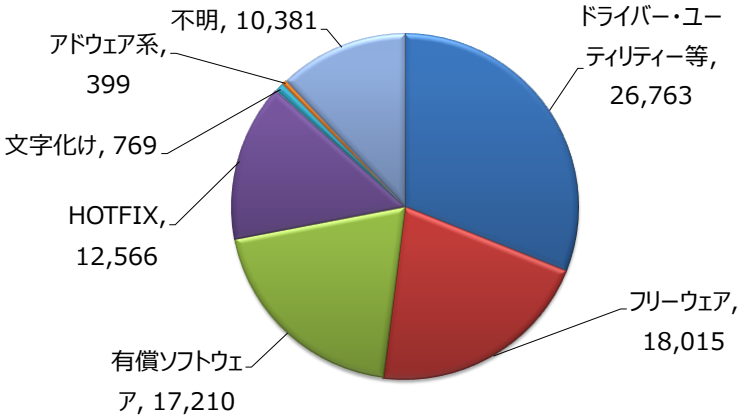


図 15　SAMAC ソフトウェア辞書に登録されているソフトウェア種別

3.3 JVN 製品データベース

JVN 脆弱性対策機械処理基盤では、JVN iPedia に脆弱性対策情報に関連する製品情報を、MyJVN に CPE に関連する製品識別子情報を格納し、これら 2 つを組み合わせて JVN 製品データベースを構成している。JVN 製品データベースに登録されている情報は、表 8 の通りであり、MyJVN API の製品一覧の API である getProductList を用いて取得することができる（図 16）。

表 8　JVN 製品データベースの登録項目

#	項目	概要	例
1	Vendor:vname	製品ベンダ名	サン・マイクロシステムズ
2	Vendor:cpe	CPE 製品ベンダ名	cpe:/:sun
3	Vendor:vid	製品ベンダ ID"	1
4	Product:pname	製品名	Sun Solaris 2.5.1 （PPC）
5	Product:cpe	CPE 製品名	cpe:/o:sun:solaris:2.5.1::ppc
6	Product:pid	製品 ID	5

```
<?xml version="1.0" encoding="UTF-8" ?>
<Result version="3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://jvndb.jvn.jp/myjvn/Results"
  xmlns:mjres="http://jvndb.jvn.jp/myjvn/Results"
  xmlns:status="http://jvndb.jvn.jp/myjvn/Status"
  xsi:schemaLocation="http://jvndb.jvn.jp/myjvn/Results http://jvndb.jvn.jp/schema/results_3.2.xsd">
      <VendorInfo xml:lang="ja">
        <Vendor vname="サン・マイクロシステムズ" cpe="cpe:/:sun" vid="1">
          <Product pname="Sun Solaris 2.5.1 （PPC）" cpe="cpe:/o:sun:solaris:2.5.1::ppc" pid="5"/>
        </Vendor>
      </VendorInfo>
      <status:Status version="3.2" method="getProductList" lang="ja" retCd="0" retMax="10000" errCd="" errMsg=""
totalRes="1" totalResRet="1" firstRes="1" productId="5" xsl="0" startItem="1"/>
</Result>
```

図 16　MyJVN API getProductList を用いた製品一覧の取得

ここでは、2014 年 11 月 10 日時点での状況について述べる。

（1）製品数

JVN 製品データベースに登録されている製品数は 22,027 件である。CPE 名（CPE 製品ベンダ名/CPE 製品名）は、製品に付与した識別子であるが、CPE 名自身の付け替えや製品ベンダの買収などにより変更することがある（表 9 **エラー! 参照元が見つかりません。**）。

表 9　JVN 製品データベースに登録されている製品数

項目	件数
製品数/CPE 製品名の件数	22,027 件

（2）製品ベンダ数

JVN 製品データベースに登録されている製品ベンダ数は、次の通りである（表 10）。

表 10　JVN 製品データベースに登録されている製品ベンダ数

項目	件数
製品ベンダ数/ CPE 製品ベンダ名の件数	10,014 件

3.4SAMAC ソフトウェア辞書と JVN 製品データベースとの紐付け

（1）調査方法

製品の紐付けについては、SAMAC ソフトウェア辞書の各エントリに、JVN 製品データベースで使用している CPE 製品名（バージョン 2.2）を追記する形式で手作業での紐付けを実施した。具体的には、9 件の既存登録項目{sw_id、sw_vendor、sw_name、sw_sname、sw_version、sw_edition、sw_alias、sw_type、sw_url}に対して、新たに登録項目{cpe}を用意し、CPE 製品名を追記した（表 11）。なお、JVN 製品データベースに登録されていない製品ベンダ、製品については、手作業で一意の製品ベンダ、製品を識別すると共に、仮の CPE 製品名を付与することで試算した。

表 11　SAMAC ソフトウェア辞書への登録項目{cpe}の追記

SAMAC ソフトウェア辞書の既存登録項目（9 項目）				連携用の新規の登録項目（1 項目）
sw_id	sw_vendor	sw_name	sw_sname, sw_version など	cpe
・・・	・・・	Adobe Acrobat 8.2.0 Professional	・・・	cpe:/a:adobe:acrobat
・・・	・・・	Adobe Acrobat 8.2.0 Standard	・・・	cpe:/a:adobe:acrobat
・・・	・・・	Adobe Acrobat 8.2.1 - CPSID_50570	・・・	cpe:/a:adobe:acrobat
・・・	・・・	Adobe Acrobat 8.2.1 Professional	・・・	cpe:/a:adobe:acrobat
・・・	・・・	Adobe Acrobat 8.2.1 Standard	・・・	cpe:/a:adobe:acrobat
・・・	・・・	Adobe Acrobat 9.3.0 - CPSID_52073	・・・	cpe:/a:adobe:acrobat

（2）製品ベンダの紐付け

SAMAC ソフトウェア辞書に登録されている辞書エントリ約 86,000 件について、JVN 製品データベースに登録されている製品ベンダとの紐付けを手作業で実施した結果を表 12 に示す。約半分の辞書エントリに対して、製品ベンダの紐付けが実施でき、104 製品ベンダ（JVN 製品データベースに登録済み＋未登録の製品ベンダを含む）が存在することと、このうち、49 製品ベンダは、JVN 製品データベースには未登録の製品ベンダであることがわかった。

表 12　SAMAC ソフトウェア辞書と JVN 製品データベースとの製品ベンダの紐付け
調査対象範囲：SAMAC ソフトウェア辞書に登録されている辞書エントリ約 86,000 件

項目		件数
製品ベンダの紐付けができなかった辞書エントリ数		42,755 件
	製品ベンダ不明の製品数（辞書エントリ数）	11,440 件
	上記以外（辞書エントリ数）	31,315 件
製品ベンダの紐付けができた辞書エントリ数		43,348 件
	うち、JVN 製品データベースに登録済みの製品ベンダ数	104 件
	うち、JVN 製品データベースに未登録の製品ベンダ数	49 件

（3）製品の紐付け

SAMAC ソフトウェア辞書に登録されている辞書エントリ 6,753 件を対象に JVN 製品データベースに登録されている製品の紐付けを手作業で実施した結果を表 13 に示す。調査対象の約半分の辞書エントリに対して、製品ベンダの紐付けが実施でき、66 製品（JVN 製品データベースに登録済み＋未登録の製品を含む）が存在することと、このうち、112 製品は、JVN 製品データベースには未登録の製品であることがわかった。

表 13　SAMAC ソフトウェア辞書と JVN 製品データベースとの製品の紐付け
調査対象範囲：SAMAC ソフトウェア辞書に登録されている辞書エントリ 6,753 件

項目		辞書エントリ数	CPE 製品名の付与件数
紐付けができなかった辞書エントリ数		3,510 件	－
紐付けができた辞書エントリ数		3,243 件	－
	うち、JVN 製品データベースに登録済みの辞書エントリ数	2,070 件	66 製品
	うち、JVN 製品データベースに未登録の辞書エントリ数	1,173 件	112 製品

（3）考察

- 紐付けができなかった製品ベンダ/製品について

  紐付けができなかった製品ベンダのうち、もともと製品ベンダの情報が格納されていないのが 26%（11,440 件）である。残り 74%（31,315 件）は、SAMAC ソフトウェア辞書の登録項目だけでは本分析期間中には特定できないと判断したものである。今後、データ連携を具体化していく中で特定を進めていくことで、特定できない比率は多少なりとも下げられると考えている。

- 紐付けができた製品ベンダ/製品のうち、JVN 製品データベースに登録済みの製品ベンダ/製品について

  本分析作業では、調査対象範囲を SAMAC ソフトウェア辞書に登録されている辞書エントリ 6,753 件に絞ってはいるが、辞書エントリの 2,070 件、製品数にすると 66 製品に対して、JVN 製品データベースで使用している CPE 製品名を追記することができた。すなわち、この 66 製品については、図 17 に示す通り、SAMAC ソフトウェア辞書を介して資産管理/インベントリ管理ツールから MyJVN API 経由で脆弱性対策情報検索することが可能となる。

  なお、脆弱性対策情報検索までの流れは、(a)資産管理ツールでインストール情報を収集し、(b)ソフトウェア名をキーとして SAMAC ソフトウェア辞書から CPE 製品名を取得した後、(c)CPE 製品名をキーとして MyJVN API でアクセスするというものである。

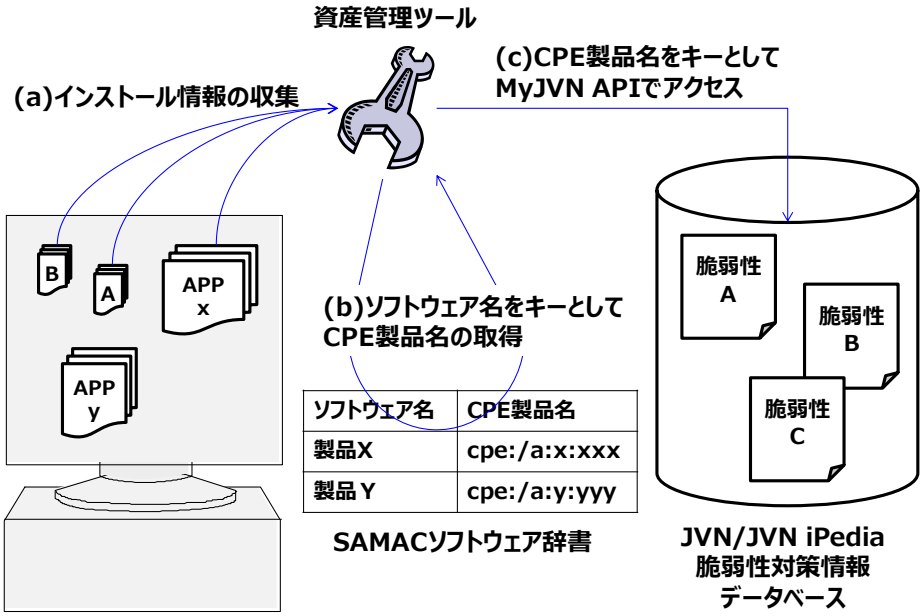


図 17　SAMAC ソフトウェア辞書と JVN/JVN iPedia 脆弱性対策情報データベースの連携

- 紐付けができた製品ベンダ/製品のうち、JVN 製品データベースに未登録の製品ベンダ/製品について

  本分析作業では、調査対象範囲を SAMAC ソフトウェア辞書に登録されている辞書エントリ 6,753 件に絞ってはいるが、辞書エントリの 1,173 件、製品数にすると 112 製品は、JVN 製品データベースに登録さ

れていない製品である。すなわち、この 112 製品については、今後、JVN 製品データベースに登録される可能性のある製品であること、JVN 製品データベースで使用している CPE 製品名がないため、SAMAC ソフトウェア辞書と JVN/JVN iPedia 脆弱性対策情報データベースの連携の対象外になってしまっている。

これら未登録製品に対しては、CPE 連携方式、SWID タグ連携方式で解決できると考えている。

- CPE 連携方式
  - ✓ 脆弱性が報告されていない製品に対しても、事前作成した CPE 製品名（CPE バージョン 2.2/2.3）を JVN 製品データベースに登録する。
  - ✓ CPE 製品名の事前作成については、NVD などの脆弱性対策情報データベースとのグローバルな運用連携を踏まえ、SWID タグから CPE 製品名を生成する手法を採用する。

- SWID タグ連携方式
  - ✓ SAMAC ソフトウェア辞書ならびに、JVN 製品データベースに SWID タグを取り込む。
  - ✓ JVN 製品データベースで SWID タグを取り込む際には、脆弱性が報告されている製品、脆弱性が報告されていない製品の双方を登録対象とする。
  - ✓ MyJVN API で SWID タグに格納されている項目、例えば、tagId をキーとして脆弱性対策情報を取得できるよう拡張する。

ただし、いずれの方式でも、国内での SWID タグの管理、グローバルでの SWID タグの連携について検討していく必要がある。

3.5 まとめ

本節では、SAMAC ソフトウェア辞書と JVN 製品データベースとの紐付けを通して、データ連携の実現方法、JVN 製品データベースに今後登録される可能性のある製品件数、JVN 製品データベースの改善課題について分析作業を通して検討した。

検討の結果、SAMAC ソフトウェア辞書と JVN/JVN iPedia 脆弱性対策情報データベースとの紐付けは可能であり、資産管理/インベントリ管理ツールを用いた脆弱性管理の実現の可能性を示した。また、JVN 製品データベースの改善課題については、JVN 製品データベースに登録されていない製品の対応にあることを示した。

4. JVN 脆弱性対策機械処理基盤での SWID タグ付与の試行

本節では、JVN 脆弱性対策機械処理基盤のツールを対象に実施した SWID タグ付与の試行について報告する。

4.1 目的

JVN/JVN iPedia 脆弱性対策情報データベースと資産管理/インベントリ管理とのデータ連携を進めるには、紐付けのための製品識別子が必要となる。特に、JVN 製品データベースに登録されていない製品の対応については、SWID タグから CPE 製品名の生成や、SWID タグに格納されている項目を利用した MyJVN API の拡張を検討する必要がある。また、NIST IR 8060 Draft3 では、SWID タグの導入、相互運用可能な SWID タグの作成について言及しているものの、英語圏以外での利用、すなわち、英語表記以外の製品ベンダ名、製品名への対応についてや、既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性確保など、運用上の課題が残っている。

そこで、JVN 脆弱性対策機械処理基盤のツールを対象に SWID タグ付与の試行を通して、次の課題について検討する。

(a)既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性を踏まえた日本語表記の製品ベンダ名、製品名への対応方法を明らかにすること

(b)SWID タグを展開する上での課題を明らかにすること

4.2 SWID タグ付与の試行

4.2.1 MyJVN 脆弱性対策情報フィルタリング収集ツール（略称：mjcheck3）

mjcheck3 は、Adobe Flash Player 版として提供している脆弱性対策情報収集ツール（略称：mjcheck）の機能拡張版で、Adobe AIR 版（略称：mjcheck3）の収集ツールである（図 18）。CPE 製品名をキーとして、製品

と JVN/JVN iPedia 脆弱性対策情報データベースとの紐付けを行うことで、該当する製品の脆弱性対策情報を収集できる。作成した SWID タグ ipa.go.jp+myjvn_mjcheck3.swidtag ファイル（図 31）の特徴は、次の通りである。

- SoftwareIdentity タグ
  - tagId は、16 バイトの GUID が推奨されている。今回、tagId は、GUID の代替案である regid＋製品＋バージョン情報を連結したテキスト形式="ipa.go.jp+myjvn_mjcheck3+3.0.0"で構成した。
- Payload タグ
  - Adobe AIR のパッケージ（mjcheck3.air）化前の mjcheck3.swf のハッシュ値と、インストール後の mjcheck3.swf のハッシュ値が一致しないことから、ipa.go.jp+myjvn_mjcheck3.swidtag ファイルへの mjcheck3.swf のハッシュ値の記載は省略した。
- その他
  - mjcheck3.swf と同じフォルダに、ipa.go.jp+myjvn_mjcheck3.swidtag ファイルがインストールされるよう Adobe AIR をパッケージ（mjcheck3.air）化した（図 19）
  - Adobe AIR のパッケージの場合、アンインストール時に ipa.go.jp+myjvn_mjcheck3.swidtag が削除されない。このため、swidtag のファイル名にバージョン情報が埋め込まれていると、mjcheck3 をバージョンアップしていく度に swidtag ファイルが作成され、古い swidtag ファイルがフォルダに残ったままとなる。このため、ipa.go.jp+myjvn_mjcheck3.swidtag ファイルには、バージョン情報を含めないファイル名とした。

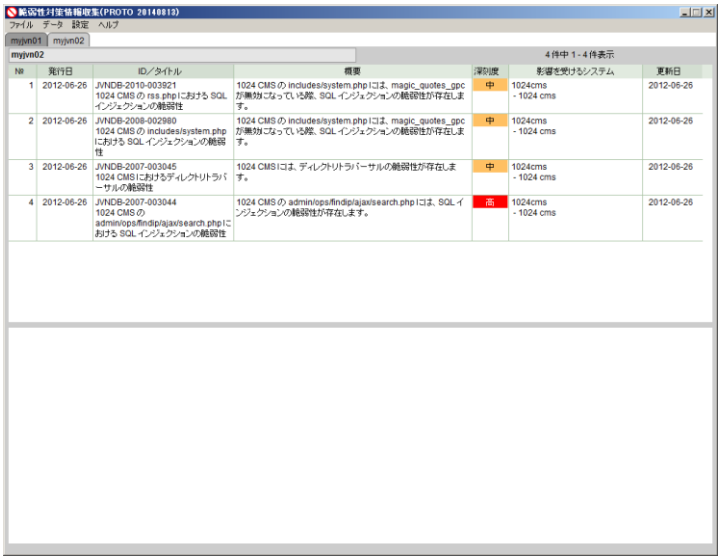


図 18　mjcheck3 の GUI

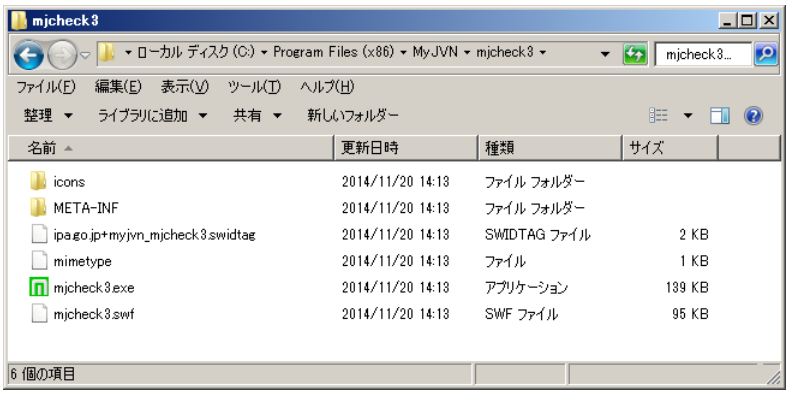


図 19　mjcheck3 のインストールフォルダ

4.2.2　MyJVN 脆弱性対策情報収集ツール（略称：mjcheck）

Adobe Flash Player 版として提供している脆弱性対策情報収集ツール（略称：mjcheck）である。作成した SWID タグ ipa.go.jp+myjvn_mjcheck.swidtag ファイル（図 32）の特徴は、次の通りである。

- SoftwareIdentity タグ
  - tagId は、16 バイトの GUID が推奨されている。今回、tagId は、GUID の代替案である regid＋製品＋バージョン情報＋言語を連結したテキスト形式＝"ipa.go.jp+myjvn_mjcheck+2.0.2+++ja"で構成した。
- Meta タグ
  - 既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性を確保しつつ、SWID タグから CPE 製品名を生成するため、Meta タグを拡張し、CPE2.3 記述を追記した（図 21）。なお、SWID タグから CPE 製品名を生成する手順は、NIST IR 8060 Draft3 で示されている手順とは異なる。
- Payload タグ
  - mjcheck.swf のハッシュ値として MD5、SHA1、SHA256 を記述した。
- Link タグ
  - mjcheck.swf と ipa.go.jp+myjvn_mjcheck.swidtag ファイルの URL を記述した。
- その他
  - Adobe Flash Player 版は、Web サイトからオンラインでダウンロードされ、ブラウザ上で実行される。mjcheck.swf 自身は、クライアント側にインストールフォルダを持たない。このため、Web サイト上の mjcheck.swf と同一フォルダに、ipa.go.jp+myjvn_mjcheck.swidtag ファイルを配備した。
  - ipa.go.jp+myjvn_mjcheck.swidtag ファイルは、http://jvndb.jvn.jp/apis/myjvn/ipa.go.jp+myjvn_mjcheck.swidtag でアクセス可能とした。また、mjcheck.swf のバージョンボタンをクリックした際には、この URL から、ipa.go.jp+myjvn_mjcheck.swidtag ファイルをダウンロードし、ダイアログ上に表示することとした（図 20）。

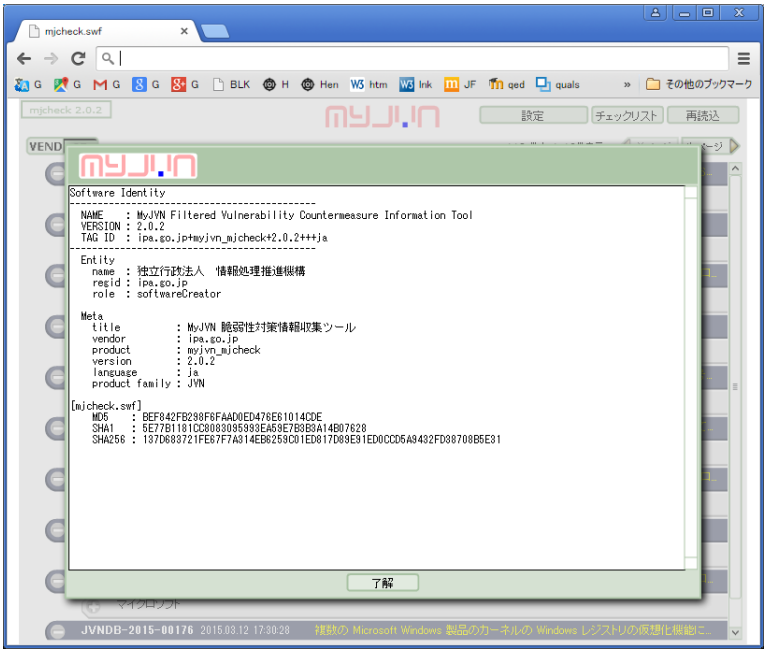


図 20　ipa.go.jp+myjvn_mjcheck.swidtag ファイルを用いたバージョン情報表示

```
<swid:Meta title="MyJVN Filtered Vulnerability Countermeasure Information Tool"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_mjcheck"
        version="2.0.2"
        update=""
        edition=""
        language="ja"
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="en"/>
```

図 21　CPE2.3 記述用の Meta タグ拡張

4.2.3　多言語版 CVSS v3 計算機（略称：ScoreCalc3）

Adobe Flash Player 版として提供している多言語版 CVSS v3 計算機（略称：ScoreCalc3）である。作成した SWID タグ ipa.go.jp+myjvn_cvss3.swidtag ファイル（図 33）の特徴は、次の通りである。

- SoftwareIdentity タグ
  - tagId は、16 バイトの GUID が推奨されている。今回、tagId は、GUID の代替案である regid＋製品＋バージョン情報を連結したテキスト形式＝"ipa.go.jp+myjvn_cvss3+1.0.0"で構成した。
- Meta タグ
  - 既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性を確保しつつ、SWID タグから CPE 製品名を生成するため、Meta タグを拡張し、CPE2.3 記述を追記した。なお、SWID タグから CPE 製品名を生成する手順は、NIST IR 8060 Draft3 で示されている手順とは異なる。
- Payload タグ
  - ScoreCalc3.swf のハッシュ値として MD5、SHA1、SHA256 を記述した。
- Link タグ
  - ScoreCalc3.swf と ipa.go.jp+myjvn_cvss3.swidtag ファイルの URL を記述した。
- その他
  - Adobe Flash Player 版は、Web サイトからオンラインでダウンロードされ、ブラウザ上で実行される。ScoreCalc3.swf 自身は、クライアント側にインストールフォルダを持たない。このため、Web サイト上の ScoreCalc3.swf と同一フォルダに、ipa.go.jp+myjvn_cvss3.swidtag ファイルを配備した。
  - ipa.go.jp+myjvn_cvss3.swidtag ファイルは、http://jvndb.jvn.jp/cvss/ipa.go.jp+myjvn_cvss3.swidtag でアクセス可能とした。また、ScoreCalc3.swf の CVSS ボタンをクリックした際には、この URL から、ipa.go.jp+myjvn_cvss3.swidtag ファイルをダウンロードし、ダイアログ上に表示することとした（図 22）。

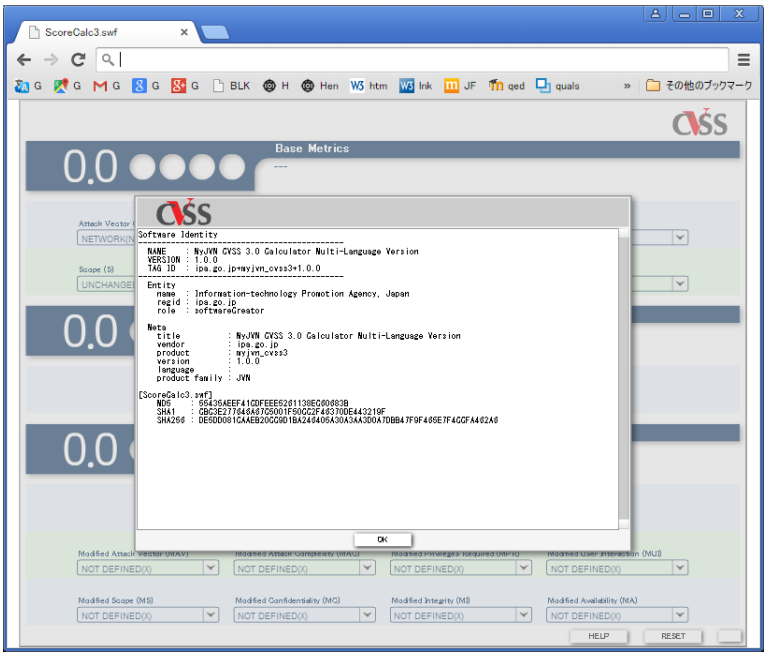


図 22　ipa.go.jp+myjvn_cvss3.swidtag ファイルを用いたバージョン情報表示

4.2.4　MyJVN バージョンチェッカ for .NET

MyJVN バージョンチェッカ for .NET は、PC にインストールされたソフトウェア製品のバージョンが最新であるかを確認するツールである。32 ビット版と 64 ビット版があり、作成した SWID タグ ipa.go.jp+myjvn_vcchecker_x32_dotnet.swidtag 、ipa.go.jp+myjvn_vcchecker_x64_dotnet.swidtag ファイル（図 34、図 35）の特徴は、次の通りである。

- SoftwareIdentity タグ
  - tagId は、16 バイトの GUID が推奨されている。今回、tagId は、GUID の代替案である regid＋製品＋バージョン情報＋言語＋32 ビット/64 ビットを連結したテキスト形式＝"ipa.go.jp+myjvn_vcchecker_x32_dotnet+1.0.0+++ja+++x32"、tagId="ipa.go.jp+myjvn_vcchecker_x64_dotnet+1.0.0+++ja+++x64"で構成した。

- Meta タグ
  - 既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性を確保しつつ、SWID タグから CPE 製品名を生成するため、Meta タグを拡張し、CPE2.3 記述を追記した。なお、SWID タグから CPE 製品名を生成する手順は、NIST IR 8060 Draft3 で示されている手順とは異なる。
- Payload タグ
  - MyJVN_.NET_CUI_x32.exe、MyJVN_.NET_GUI_x32.exe、MyJVN_.NET_CUI_x64.exe、MyJVN_.NET_GUI_x64.exe のハッシュ値として MD5、SHA1、SHA256 を記述した。
- Link タグ
  - MyJVN バージョンチェッカ for .NET で使用している「Microsoft .NET Framework」コンポーネントの CPE 製品名を記述した。
- その他
  - MyJVN_.NET_CUI_x32.exe、MyJVN_.NET_GUI_x32.exe、MyJVN_.NET_CUI_x64.exe、MyJVN_.NET_GUI_x64.exe と同じフォルダに、swidtag ファイルがインストールされるようパッケージ化した（図 23、図 24）

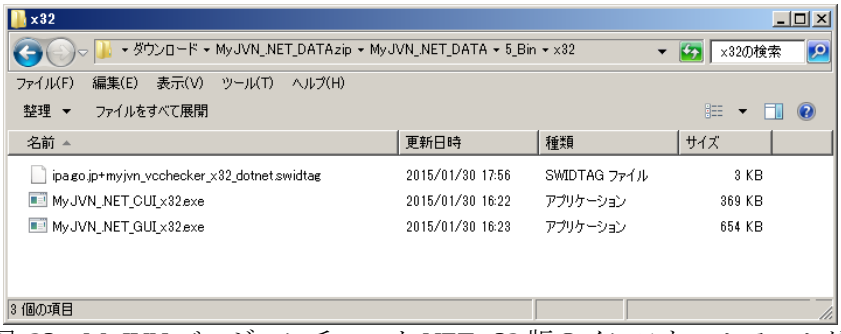


図 23　MyJVN バージョンチェッカ.NET x32 版のインストールフォルダ

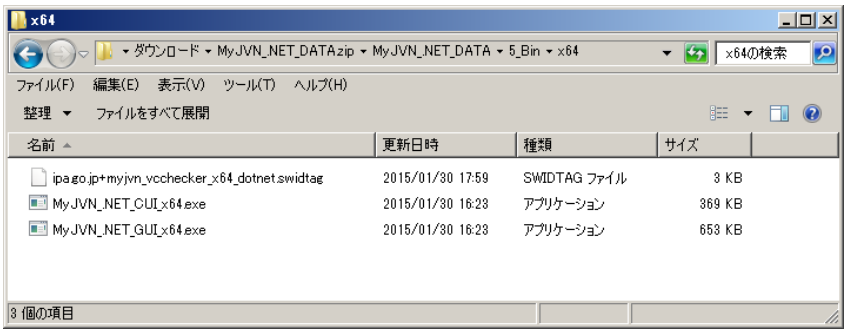


図 24　MyJVN バージョンチェッカ.NET x64 版のインストールフォルダ

4.3 日本語表記の SWID タグの作成について

4.4 要件を満たす日本語表記の SWID タグ

SWID タグ付与の試行と 2015 年 8 月にリリースされた NIST IR 8060 Draft3 を踏まえて、下記 3 つの要件を満たす日本語表記の SWID タグの作成について検討した結果について述べる。
(a)日本語と英語の併記への対応
(b)NIST IR 8060 Draft3 との整合性の確保
(c)既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性の確保

図 25 に、上記 3 つの要件を満たす日本語表記の SWID タグを示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
```

```
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="MyJVN Filtered Vulnerability Countermeasure Information Tool 3"
    version="3.0.0"
    tagId="559c90d5-68ba-4d6e-80ab-fba3f59bbcef"
    versionScheme="multipartnumeric"
    >
    <swid:Entity ・・・(c)既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性の確保
                          (b)NIST IR 8060 Draft3 との整合性の確保
        name="ipa"
        regid="ipa.go.jp"
        role="softwareCreator"/>
    <swid:Entity ・・・(a)日本語と英語の併記への対応
        name="Information-technology Promotion Agency, Japan"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="en"/>
    <swid:Entity ・・・(a)日本語と英語の併記への対応
        name="独立行政法人　情報処理推進機構"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta ・・・(c)既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性の確保
                          (b)NIST IR 8060 Draft3 との整合性の確保
        product="myjvn"
        colloquialVersion="api"
        revision="update3"
        edition="ed"/>
    <swid:Meta ・・・(a)日本語と英語の併記への対応
        product="MyJVN Filtered Vulnerability Countermeasure Information Tool 3"
        colloquialVersion="Application Program Interface"
        revision="Uppdate Version 3"
        edition="Edition"
        xml:lang="en"/>
    <swid:Meta ・・・(a)日本語と英語の併記への対応
        product="MyJVN 脆弱性対策情報フィルタリング収集ツール"
        colloquialVersion="アプリケーションプログラムインタフェース"
        revision="アップデートバージョン 3"
        edition="エディション"
        xml:lang="ja"/>
    <swid:Payload>
          :
    </swid:Payload>
    <swid:Link href="cpe:/a:openssl:openssl" rel="related"/>
          :
</swid:SoftwareIdentity>
```

図 25　3 つの要件を満たす日本語表記の SWID タグ

4.4.1　課題

3 つの要件を満たす日本語表記の SWID タグ（図 25）に関する課題は、次の通りである。

- NIST IR 8060 Draft3 では、Entity タグ、Meta タグが複数記載されていることを想定していない。SWID タグから CPE バージョン 2.3 の生成を手順するためには、どの Entity タグ、Meta タグを使用するのかが明確になっていなければならない。

4.5 まとめ

本節では、日本語表記の SWID タグ、SWID タグを展開する上での課題について分析作業を通して検討した。

検討の結果、SWID から CPE 製品名を生成するための Entity/Meta タグ、日本語表記用の Entity/Meta タグ、英語表記用の Entity/Meta タグを用意することで実現できる可能性を示した。

5. 資産管理と脆弱性管理の連携の試行

本節では、SWID タグファイルの活用範囲の拡大として、SWID タグファイルを用いたユーザアプリケーション関連の資産管理と脆弱性管理の連携について報告する。

5.1 目的

ユーザアプリケーション開発では、Apache Struts のようなフレームワークや OpenSSL などのライブラリが利用されることも多い。その一方で、納品物であるユーザアプリケーションに、どのような外部コンポーネントが組み込まれているか、前提プログラムが何であるのかまでを管理する仕組みが整備されているわけではない。このため、2014 年に報告された Apache Struts、Heartbleed などの脆弱性が報告された場合、影響を受けるコンポーネントを使用しているかどうかは、開発業者に問い合わせない限り判断できないという状況につながっている。

ユーザアプリケーションの納品において、ユーザアプリケーションだけではなく、使用している外部コンポーネントや前提プログラムを把握できるユーザアプリケーション用 SWID タグファイルを納品してもらうことができれば、ユーザアプリケーション自身についても資産管理だけではなく、脆弱性管理も可能となる。

そこで、ユーザアプリケーション開発における次の課題について検討する。
(a)ユーザアプリケーション用 SWID タグに必要となる項目を明らかにすること
(b)ユーザアプリケーションを対象とする資産管理と脆弱性管理の連携の実現方法を明らかにすること

5.2 ユーザアプリケーション用 SWID タグの試作

ユーザアプリケーションとして開発している JVN iPedia、MyJVN を対象に作成したユーザアプリケーション用 SWID タグ ipa.go.jp+ipedia.swidtag、ipa.go.jp+myjvn.swidtag ファイル（図 36、図 37）の特徴は、次の通りである。

- SoftwareIdentity タグ
  - tagId は、16 バイトの GUID が推奨されている。今回、tagId は、GUID の代替案である regid＋製品＋バージョン情報を連結したテキスト形式＝"ipa.go.jp+jvn_ipedia+3.5.0"、"ipa.go.jp+myjvn_api+3.2.0"で構成した。
- Meta タグ
  - 既存の CPE 名（CPE 製品ベンダ名/CPE 製品名）との整合性を確保しつつ、SWID タグから CPE 製品名を生成するため、Meta タグを拡張し、CPE2.3 記述を追記した（図 21）。なお、SWID タグから CPE 製品名（CPE バージョン 2.3）を生成する手順は、NIST IR 8060 で示されている手順とは異なる。
- Payload タグ
  - ユーザアプリケーションの主要モジュールである index.php、MyJVN.class のハッシュ値として MD5、SHA1、SHA256 を記述した。
- Link タグ
  - 使用している外部コンポーネントや前提プログラムの CPE 製品名（バージョン 2.2）を記述した（図 26）。

```
<swid:Link href="cpe:/a:oracle:jre" rel="related"/>
<swid:Link href="cpe:/a:openssl:openssl" rel="related"/>
<swid:Link href="cpe:/a:openssh:openssh" rel="related"/>
<swid:Link href="cpe:/a:ntp:ntp" rel="related"/>
<swid:Link href="cpe:/a:apache:apr-util" rel="related"/>
<swid:Link href="cpe:/a:pcre:pcre" rel="related"/>
<swid:Link href="cpe:/a:apache:http_server" rel="related"/>
<swid:Link href="cpe:/a:apache:tomcat" rel="related"/>
<swid:Link href="cpe:/a:mysql:mysql" rel="related"/>
```

図 26　外部コンポーネントや前提プログラムの CPE 製品名記述

Link タグに記載した使用している外部コンポーネントや前提プログラムの CPE 製品名を用いた脆弱性対策情報検索の流れは、(a)納品物として納品されたユーザアプリケーション用 SWID タグを資産管理ツールに取り

込み、(b)Link タグに記載されている CPE 製品名を抽出した後、脆弱性管理ツールに通知する。(c)脆弱性管理ツールは、CPE 製品名をキーとして MyJVN API でアクセスするというものである（図 27）。

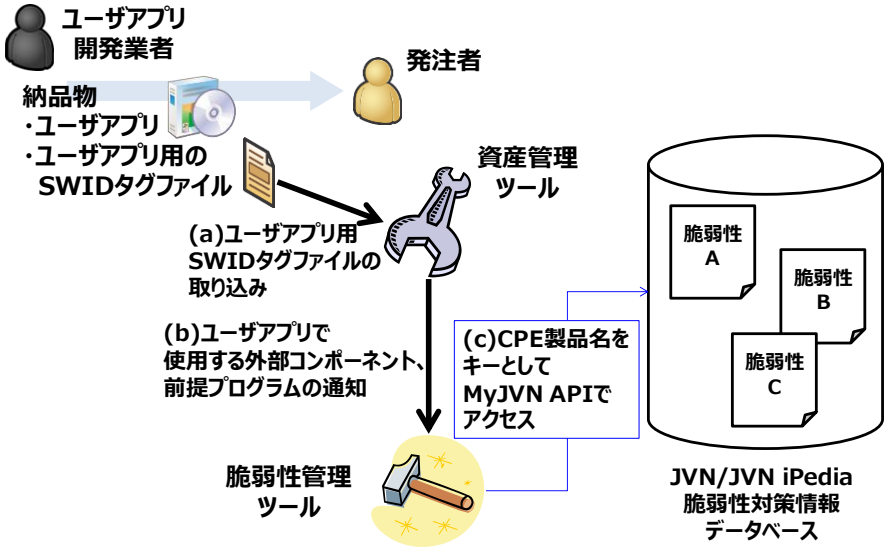


図 27　ユーザアプリケーション用 SWID タグファイルを利用した脆弱性対策情報検索の概要

5.3 mjcheck3 の機能拡張プロトタイプの試作

既存版 mjcheck3 は、CPE 製品名を手入力することで、製品と JVN/JVN iPedia 脆弱性対策情報データベースとのとの紐付けを行っている。これにより、JVN/JVN iPedia 脆弱性対策情報データベースから該当する製品の脆弱性対策情報のみを取得している。機能拡張プロトタイプである試作版 mjcheck3 では、SWID タグファイルをインポートすることにより、製品と JVN/JVN iPedia 脆弱性対策情報データベースとの紐付けを行う（図 28）。これにより、発注者側での資産管理や脆弱性管理ツールへの手入力の工数をなくすことができ、さらに、Link タグに記載されている外部コンポーネントや前提プログラムの CPE 製品名を用いて脆弱性対策情報検索が可能となる。

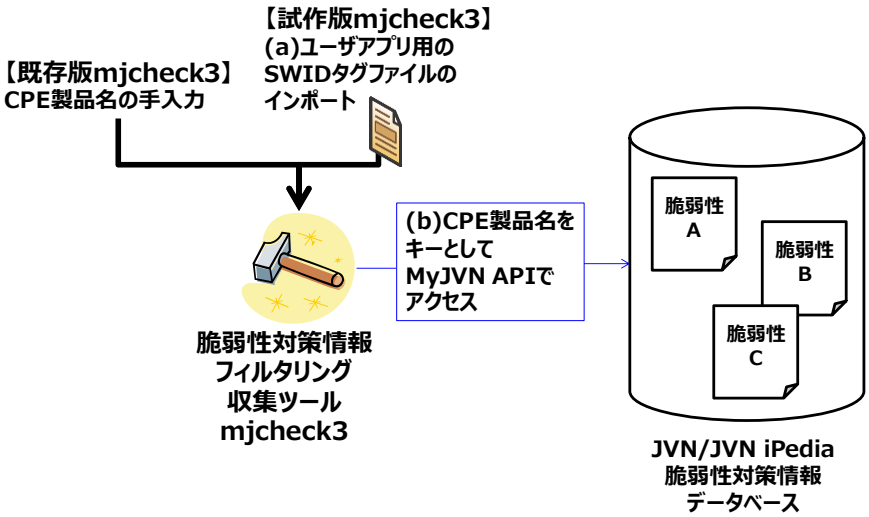


図 28　mjcheck3 の機能拡張プロトタイプによる脆弱性管理の流れ

試作版 mjcheck3 で実装した SWID タグファイルをインポートの GUI 操作を図 29、図 30 に示す。図 29 は、インポートする SWID タグファイルを指定するまで操作、図 30 は、インポートした SWID タグファイルに含まれている製品一覧、すなわち、Link タグに記載されている外部コンポーネントや前提プログラムを表示している操作である。

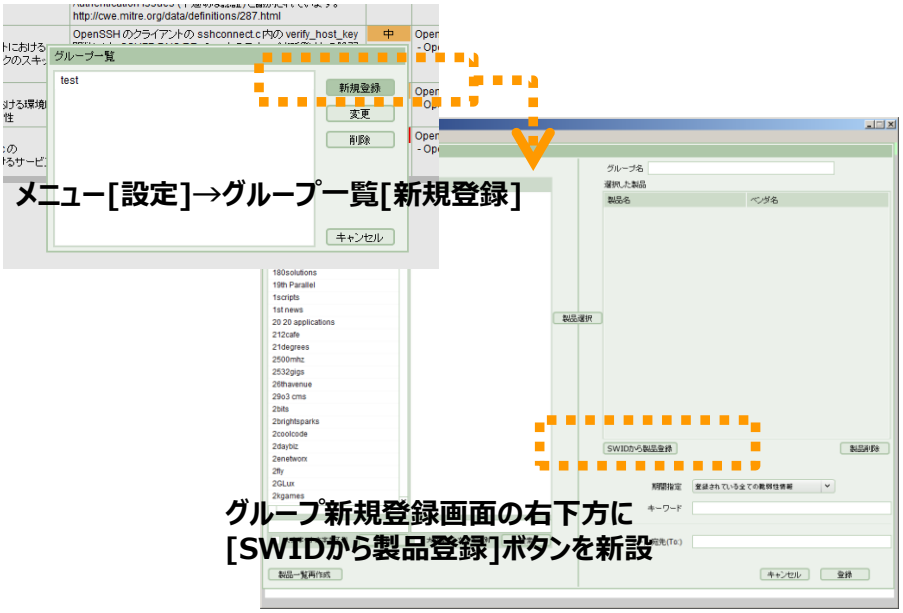


図 29　SWID タグファイルのインポート機能

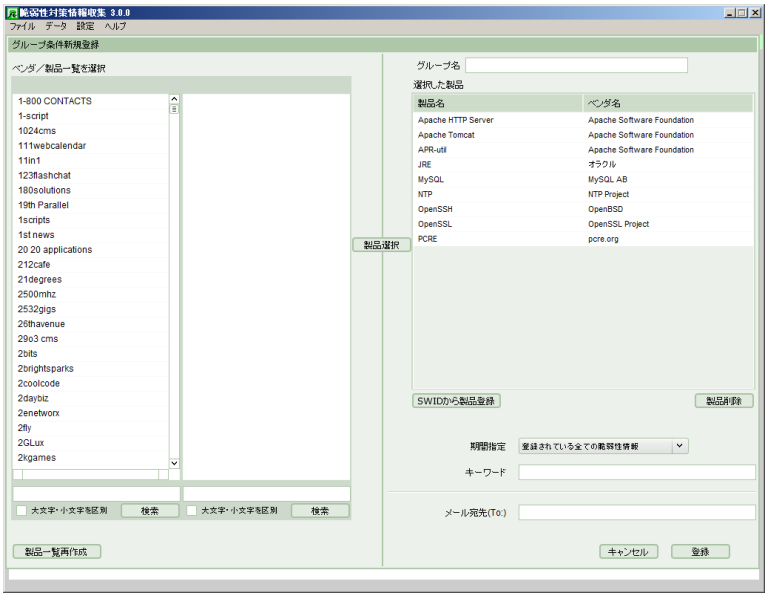


図 30　　SWID タグファイルのインポートによる製品登録

5.4 まとめ

本節では、ユーザアプリケーション用 SWID タグファイルを利用した、ユーザアプリケーション関連の資産管理と脆弱性管理の連携について分析作業を通して検討した。

検討の結果、ユーザアプリケーション用 SWID タグファイルの中に、Link タグを用いて外部コンポーネントや前提プログラムの CPE 製品名を記述することで、ユーザアプリケーションならびに、付随するプログラムを資産管理の一部として管理できる可能性を示した。また、Link タグのデータを利用することで、ツールを用いた脆弱性管理の実現の可能性を示した。

6. 付録

本節では、参考情報として試行で作成した SWID タグを示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="MyJVN Filtered Vulnerability Countermeasure Information Tool 3"
    version="3.0.0"
    tagId="ipa.go.jp+myjvn_mjcheck3+3.0.0"
    versionScheme="multipartnumeric"
    >
    <swid:Entity
        name="Information-technology Promotion Agency, Japan"
        regid="http://ipa.go.jp"
        role="softwareCreator"/>
    <swid:Entity
        name="独立行政法人　情報処理推進機構"
        regid="http://ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta product="MyJVN Filtered Vulnerability Countermeasure Information Tool 3"
        version="3.0.0"
        productFamily="JVN"/>
    <swid:Meta product="MyJVN 脆弱性対策情報フィルタリング収集ツール"
        version="3.0.0"
        productFamily="JVN"
        xml:lang="ja"/>
    <swid:Payload>
        <swid:File name="mjcheck3.exe"/>
        <swid:File name="mjcheck3.swf"/>
    </swid:Payload>
</swid:SoftwareIdentity>
```

図 31　ipa.go.jp+myjvn_mjcheck3.swidtag

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="MyJVN Filtered Vulnerability Countermeasure Information Tool"
    version="2.0.2"
    tagId="ipa.go.jp+myjvn_mjcheck+2.0.2+++ja"
    versionScheme="multipartnumeric"
    >
    <swid:Entity
        name="Information-technology Promotion Agency, Japan"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="en"/>
    <swid:Entity
        name="独立行政法人　情報処理推進機構"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta title="MyJVN Filtered Vulnerability Countermeasure Information Tool"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_mjcheck"
        version="2.0.2"
        update=""
        edition=""
        language="ja"
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
```

```
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="en"/>
    <swid:Meta title="MyJVN 脆弱性対策情報収集ツール"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_mjcheck"
        version="2.0.2"
        update=""
        edition=""
        language="ja"
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="ja"/>
    <swid:Payload>
        <swid:File name="mjcheck.swf"
            MD5:hash="BEF842FB298F6FAAD0ED476E61014CDE"
            SHA1:hash="5E77B1181CC8083095993EA59E7B3B3A14B07628"
            SHA256:hash="137D683721FE67F7A314EB6259C01ED817D89E91ED0CCD5A9432FD38708B5E31"/>
    </swid:Payload>
    <swid:Link href="http://jvndb.jvn.jp/apis/myjvn/mjcheck.swf" rel="component" type="application/vnd.adobe.flash-movie" />
    <swid:Link href="http://jvndb.jvn.jp/apis/myjvn/ipa.go.jp+myjvn_mjcheck.swidtag" rel="swidtag" type="application/xml" />
</swid:SoftwareIdentity>
```

図 32　http://jvndb.jvn.jp/apis/myjvn/ipa.go.jp+myjvn_mjcheck.swidtag

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="MyJVN CVSS 3.0 Calculator Multi-Language Version"
    version="1.0.0"
    tagId="ipa.go.jp+myjvn_cvss3+1.0.0"
    versionScheme="multipartnumeric"
    >
    <swid:Entity
        name="Information-technology Promotion Agency, Japan"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="en"/>
    <swid:Entity
        name="独立行政法人　情報処理推進機構"
        regid="http://ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta title="MyJVN CVSS 3.0 Calculator Multi-Language Version"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_cvss3"
        version="1.0.0"
        update=""
        edition=""
        language=""
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="en"/>
    <swid:Meta title="MyJVN CVSS 計算ソフトウェア 多国語版"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_cvss3"
        version="1.0.0"
        update=""
        edition=""
        language=""
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
```

```
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="ja"/>
    <swid:Payload>
        <swid:File name="ScoreCalc3.swf"
            MD5:hash="55435AEEF41CDFEEE5261138EC60683B"
            SHA1:hash="CBC3E277646A67C5001F50CC2F46370DE443219F"
            SHA256:hash="DE5DD081CAAEB20CC9D1BA246405A30A3AA3D0A7DBB47F9F465E7F4CCFA462A6"/>
    </swid:Payload>
    <swid:Link href="http://jvndb.jvn.jp/cvss/ScoreCalc3.swf" rel="component" type="application/vnd.adobe.flash-movie" />
    <swid:Link href="http://jvndb.jvn.jp/cvss/ipa.go.jp+myjvn_cvss3.swidtag" rel="swidtag" type="application/xml" />
</swid:SoftwareIdentity>
```

図 33　http://jvndb.jvn.jp/cvss/ipa.go.jp+myjvn_cvss3.swidtag

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="MyJVN Version Checker (32-bit) for .NET"
    version="1.0.0"
    tagId="ipa.go.jp+myjvn_vcchecker_x32_dotnet+1.0.0+++ja+++x32"
    versionScheme="multipartnumeric"
    >
    <swid:Entity
        name="Information-technology Promotion Agency, Japan"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="en"/>
    <swid:Entity
        name="独立行政法人　情報処理推進機構"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta title="MyJVN Version Checker (32-bit) for .NET"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_vcchecker_x32_dotnet"
        version="1.0.0"
        update=""
        edition=""
        language="ja"
        sw_edition=""
        target_sw=""
        target_hw="x32"
        productFamily="JVN"
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="en"/>
    <swid:Meta title="MyJVN バージョンチェッカ (32-bit) for .NET"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_vcchecker_x32_dotnet"
        version="1.0.0"
        update=""
        edition=""
        language="ja"
        sw_edition=""
        target_sw=""
        target_hw="x32"
        productFamily="JVN"
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="ja"/>
    <swid:Payload>
        <swid:File name="MyJVN_.NET_CUI_x32.exe"
            MD5:hash="88e110e169b9324b02db6b824571a432"
            SHA1:hash="6faa45fab22ef156f3c07c0a84e0bf30c2c6f2b7"
            SHA256:hash="dcf90e32955af68a53767a1785d164fb955a81279e2acb6f229b92a309d1f4ac"/>
        <swid:File name="MyJVN_.NET_GUI_x32.exe"
            MD5:hash="83468b0539f8201380b797ec735a4309"
            SHA1:hash="3cb58336a7c802fb11113584e66a65b0efc661fa"
            SHA256:hash="16f4ba701ee336269a9763fca3bb51f2f4aa506346eab6233ce5224e94c67756"/>
    </swid:Payload>
    <swid:Link href="cpe:/a:microsoft:.net_framework" rel="component"/>
</swid:SoftwareIdentity>
```

図 34　ipa.go.jp+myjvn_vcchecker_x32_dotnet.swidtag

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="MyJVN Version Checker (64-bit) for .NET"
    version="1.0.0"
    tagId="ipa.go.jp+myjvn_vcchecker_x64_dotnet+1.0.0+++ja+++x64"
    versionScheme="multipartnumeric"
    >
    <swid:Entity
        name="Information-technology Promotion Agency, Japan"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="en"/>
    <swid:Entity
        name="独立行政法人　情報処理推進機構"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta title="MyJVN Version Checker (64-bit) for .NET"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_vcchecker_x64_dotnet"
        version="1.0.0"
        update=""
        edition=""
        language="ja"
        sw_edition=""
        target_sw=""
        target_hw="x64"
        productFamily="JVN"
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="en"/>
    <swid:Meta title="MyJVN バージョンチェッカ (64-bit) for .NET"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_vcchecker_x64_dotnet"
        version="1.0.0"
        update=""
        edition=""
        language="ja"
        sw_edition=""
        target_sw=""
        target_hw="x64"
        productFamily="JVN"
        timestamp="2015-01-26T00:00:00+09:00"
        xml:lang="ja"/>
    <swid:Payload>
        <swid:File name="MyJVN_.NET_CUI_x64.exe"
            MD5:hash="24fff5d54825a290aaa25017e8adc8cd"
            SHA1:hash="33d2d673c65a5f84c68edbd16ab2cd43401f0478"
            SHA256:hash="7483111831b9394eb771c14f46d21bf0acc7f577dc63181cb876da46ddef5e85"/>
        <swid:File name="MyJVN_.NET_GUI_x64.exe"
            MD5:hash="584ee5f650f9b45777d4d8805345b831"
            SHA1:hash="0ed86be78ee214d50c49a65691a905062a9e436e"
            SHA256:hash="a58315e6557b41029952cfe49f85d7f7e1ea4b59af3d8f95acdfb7c527a987d6"/>
    </swid:Payload>
    <swid:Link href="cpe:/a:microsoft:.net_framework" rel="component"/>
</swid:SoftwareIdentity>
```

図 35　ipa.go.jp+myjvn_vcchecker_x64_dotnet.swidtag

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="JVN iPedia Web Application"
```

```
    version="3.5.0"
    tagId="ipa.go.jp+jvn_ipedia+3.5.0"
    versionScheme="multipartnumeric"
    >
    <swid:Entity
        name="Information-technology Promotion Agency, Japan"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="en"/>
    <swid:Entity
        name="独立行政法人　情報処理推進機構"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta title="JVN iPedia"
        part="a"
        vendor="ipa.go.jp"
        product="jvn_ipedia"
        version="3.5.0"
        update=""
        edition=""
        language=""
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
        timestamp="2015-02-06T00:00:00+09:00"
        xml:lang="en"/>
    <swid:Meta title="JVN iPedia"
        part="a"
        vendor="ipa.go.jp"
        product="jvn_ipedia"
        version="3.5.0"
        update=""
        edition=""
        language=""
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
        timestamp="2015-02-06T00:00:00+09:00"
        xml:lang="ja"/>
    <swid:Payload>
        <swid:File name="index.php"
            MD5:hash="54e0197f9a490b90f2692d6897348a36"
            SHA1:hash="b81fc33ee6fbe0d3d7a716d89417841c865ce5d2"
            SHA256:hash="e6091530d7de65b072d91c6f6f447a368a5c60535504f7f665bbf624dc58592f"/>
    </swid:Payload>
    <swid:Link href="cpe:/a:openssl:openssl" rel="related"/>
    <swid:Link href="cpe:/a:openssh:openssh" rel="related"/>
    <swid:Link href="cpe:/a:ntp:ntp" rel="related"/>
    <swid:Link href="cpe:/a:apache:apr-util" rel="related"/>
    <swid:Link href="cpe:/a:pcre:pcre" rel="related"/>
    <swid:Link href="cpe:/a:apache:http_server" rel="related"/>
    <swid:Link href="cpe:/a:php:php" rel="related"/>
    <swid:Link href="cpe:/a:mysql:mysql" rel="related"/>
</swid:SoftwareIdentity>
```

図 36　ipa.go.jp+ipedia.swidtag

```
<?xml version="1.0" encoding="UTF-8"?>
<swid:SoftwareIdentity
    xmlns:swid="http://jvndb.jvn.jp/schema/swid.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:MD5="http://www.w3.org/2001/04/xmldsig-more#md5"
    xmlns:SHA1="http://www.w3.org/2000/09/xmldsig#sha1"
    xmlns:SHA256="http://www.w3.org/2001/04/xmlenc#sha256"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xsi:schemaLocation="http://jvndb.jvn.jp/schema/swid.xsd http://jvndb.jvn.jp/schema/swid.xsd"
    name="MyJVN API"
    version="3.2.0"
    tagId="ipa.go.jp+myjvn_api+3.2.0"
    versionScheme="multipartnumeric"
    >
    <swid:Entity
        name="Information-technology Promotion Agency, Japan"
        regid="ipa.go.jp"
        role="softwareCreator"
```

```
        xml:lang="en"/>
    <swid:Entity
        name="独立行政法人　情報処理推進機構"
        regid="ipa.go.jp"
        role="softwareCreator"
        xml:lang="ja"/>
    <swid:Meta title="MyJVN API"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_api"
        version="3.2.0"
        update=""
        edition=""
        language=""
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
        timestamp="2015-02-06T00:00:00+09:00"
        xml:lang="en"/>
    <swid:Meta title="MyJVN API"
        part="a"
        vendor="ipa.go.jp"
        product="myjvn_api"
        version="3.2.0"
        update=""
        edition=""
        language=""
        sw_edition=""
        target_sw=""
        target_hw=""
        productFamily="JVN"
        timestamp="2015-02-06T00:00:00+09:00"
        xml:lang="ja"/>
    <swid:Payload>
        <swid:File name="MyJVN.class"
            MD5:hash="21c4dd282ce97ea67da7e08f31311ba2"
            SHA1:hash="750e9e7dfd57b60a0cb9114cc5f88a9ba82f14cc"
            SHA256:hash="86ce76595fa4ff0e487079206178ddc9a2e37c58b5232d4a4af0442805cb37c2"/>
    </swid:Payload>
    <swid:Link href="cpe:/a:oracle:jre" rel="related"/>
    <swid:Link href="cpe:/a:openssl:openssl" rel="related"/>
    <swid:Link href="cpe:/a:openssh:openssh" rel="related"/>
    <swid:Link href="cpe:/a:ntp:ntp" rel="related"/>
    <swid:Link href="cpe:/a:apache:apr-util" rel="related"/>
    <swid:Link href="cpe:/a:pcre:pcre" rel="related"/>
    <swid:Link href="cpe:/a:apache:http_server" rel="related"/>
    <swid:Link href="cpe:/a:apache:tomcat" rel="related"/>
    <swid:Link href="cpe:/a:mysql:mysql" rel="related"/>
</swid:SoftwareIdentity>
```

図 37　ipa.go.jp+myjvn.swidtag

第４章　ソフトウェア識別情報の標準化動向

1. 背景

プラントの制御ソフトウェアや組み込みソフトウェアなどの制御ソフトウェアでは、IEC 61508 に代表されるセーフティのための標準化に関して長い歴史がある、一方、ソフトウェアセキュリティについては、物理的な隔離が前提となっていることが多かった。

逆に、アプリケーションソフトウェアの世界では、様々なセキュリティ対策が検討される一方、セーフティについて議論されることは少なかった。

しかし、家電製品、自動車、ロボットなどの制御ソフトウェアがネットワークに接続され、アプリケーションソフトウェアと連動して動作するようになるにともない、制御ソフトウェアとアプリケーションソフトウェアには明確な区別がなくなりつつある。そのため、システム全体としてセキュリティ及びセーフティの両面から新たな対策の必要性が叫ばれるようになった。

近年、様々なデバイスへの対応の必要性などから、HTML、JSON などプラットフォーム提供会社に依存しない標準的な技術を利用したソフトウェアの開発が一般的となっている。このため、アプリケーションソフトウェアの開発にあたり、サードパーティ製やオープンソフトウェアとして配布されているライブラリやコンポーネントを利用する機会が多くなり、このことがソフトウェア開発や脆弱性への対応などの管理を難しくしている。

加えて、ソフトウェア開発及びハードウェア開発のグローバル化が進み安全保障上の懸念を耳にすることも多くなってきている。

一方で、ソフトウェアの管理が独占や寡占、または、新たな貿易障壁などの国際経済的な問題となる懸念もある。

2. 標準化動向

このような背景の中、ここ数年、米国（特に、米国国防総省）が中心となり、ソフトウェアの出所管理を行うための標準化活動が模索されている。

2.1 2009 年 ISO/IEC 19770-2 「Software identification tag」

ソフトウェアを識別するための国際規格としては、ISO/IEC JTC 1 SC7 WG21 により開発された ISO/IEC 19770-2 「Software identification tag」が 2009 年に制定されている。しかし、この国際規格は、ソフトウェア資産管理を目的とした規格であり、セキュリティ対策を目的としたソフトウェアの出所管理などはスコープ外であった。

技術的にはコンポーネントやファイル単位での管理も可能であったが、ライセンス単位の管理を行うことを中心とした設計となっていたため、必ずしもコンポーネントやファイル単位での管理に適したものではなかった。

2.2 2014 年 ISO/IEC 17960 「Code signing for source code」Draft International Standard (DIS)

ISO/IEC SC22 WG23 により、ソフトウェアの出所を管理するため、開発者によるソースプログラムへ署名の標準化として2010年から開発が開始される。編集履歴の管理までも視野に入れた規格であり、当初はプログラム言語に依存しない API の標準化を目指していた。しかし、既にコード署名を行うためのソフトウェアが市場に存在すること、管理の粒度が細かすぎることやプログラム言語に依存しない API の規定が困難であることなどから、最終的には、コード署名の要件のみを規定した１０ページ程度の（情報分野としては）非常に小さな規格として 2014 年に DIS が承認された。

日本は、セーフティ、セキュリティの両面から出所管理を可能とする規格として着目し、API の要件定義を盛り込むことを提案し、将来版にて対応すること条件に賛成した。しかし、国際規格成立後、米国人の議長が突如スポンサーを切られるとともに、米国から WG の終了を問う投票が提案されるなどの騒動があり、最終的に WG は存続することになったが、ISO/IEC 17960 の更新版は作成されないことが決定した。

2.3 201x 年 ISO/IEC 19770-2 「Software identification tag」Working Draft

2014 年初めに開発が開始された ISO/IEC 19770-2:2009 の更新版。ISO/IEC 19770-2:2009 から大きくスコープを広げ、セキュリティの観点からのソフトウェアの管理や自動的な脆弱性の発見など、セキュリティ対策への活用が大きく歌われた規格となっている。

コンポーネントやファイル単位での出所管理に必要な規定が強化され、ソフトウェアの構成要素の情報の包括的ツリーを構築することを重視している。さらに、各種デバイスに対するソフトウェア ID タグの付与についても、その方法に至るまで言及するなど、徹底した出所管理を行うこと想定した規格となっている。

現在 DIS Ballot の最中であり、順調に進めば、2016 年前半頃には、国際規格として成立しそうだ。

2.4 欧州委員会の懸念

2012 年 1 月、欧州委員会の DG Connect と DG Enterprise が共同で、MSP(Multi-stakeholder platform on ICT standardization)を設立した。MSP は、欧州の Digital Agenda を実現するために設けられた検討組織の一つであり、メンバ国の IT ポリシ担当者及び民間標準化団体の代表者から構成され、情報システムに係る標準化が欧州の経済に与える影響について検討するとともに、その標準化の視点からの IT 戦略、調達ポリシなど検討するグループである。政府調達で採用すべき技術のリストを作ることなどが初期の主要な議題となっている。

最近、MSP は、公な文書などで表明したわけではないが、関係者内で議論されていることとして、「安全性」の名の下に進む独占化への懸念が示されている。

懸念する標準技術としては、UEFI（Universal Extended Firmware Interface） と HTML5 が挙げられる

（図 38）。

UEFI は、ハードウェアと OS との間に位置する、いわゆる BIOS に相当するソフトウェアの規格であるが、ウィルスやマルウェアなどによる脅威を回避するために、電子署名されたソフトウェア（デバイスドライバや OS などを含む）以外動作しない様に制御する機能が規格化されている。

この規格そのものはオープンであるが、運用上は、チップの上にあらかじめ電子署名を確認するために必要となる公開鍵をインストールしておく必要があり、その公開鍵に対応する秘密鍵を持っている者だけが、ソフトウェアに署名を行うことができることとなる。

結果として、現在流通している一般の PC の上で動作するデバイスドライバ等を出荷するためには、特定の者にそのソースプログラム等を示して検証を受けた上で、電子署名をもらう必要が出てきている。このように、鍵の管理が事実上一つの企業に集中しつつある、というのが MSP の懸念である。

HTML 5 は、ウェブの新しい規格であるが、その新機能として、コンテンツの著作権管理機能が盛り込まれている。著作権管理を行うためには、ブラウザが、著作権管理ポリシに従った正しい動作をすること（不正なコピー等を行えないような対策を取るなど）が保証される必要があるが、このためには、ブラウザが動作を確認して認定し、それに対して電子署名を発行するといった運用が必要となる。この運用に公平さが欠けると、ブラウザソフトウェアの独占化が進のではないか、というのが MSP 関係者の懸念である。

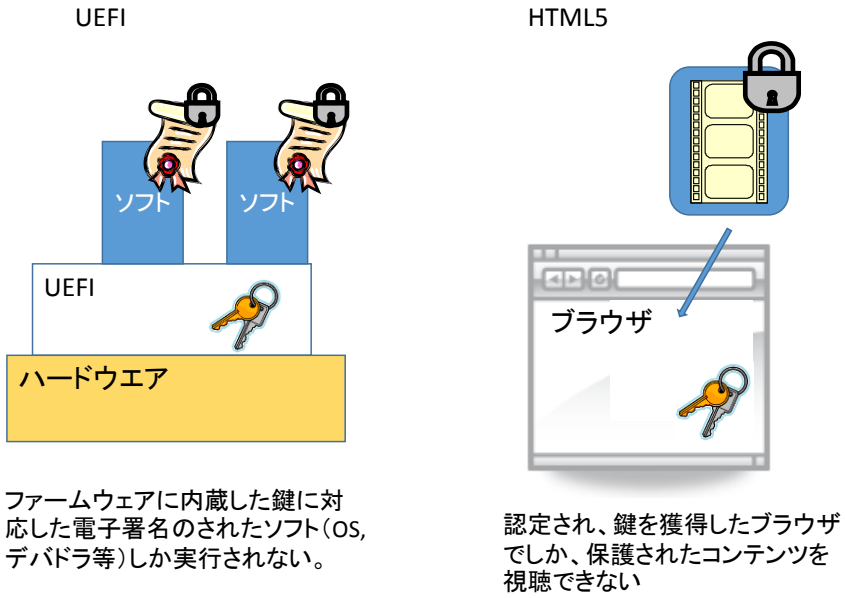


図 38　懸念される 2 つの標準技術

このような懸念を解消するためには、ソフトウェアを確認し、認定する第三者的な公平な機関を設け、そこが証明書を発行したソフトウェアについては原則無条件で UEFI や HTML 5 の枠組みの中で動作するように運用すると行ったことが必要であると考えられる。

このような観点からは、ソフトウェアの識別法、そこへのタグ付け（認定した証明などを含む）法、タグとソフトウェア本体との一致性の保証方法（電子署名法）及びその公平な運用法の確立が必要である。

3. まとめ

ソフトウェアの識別・管理に関する標準化は、ここ数年比較的早い速度で進められている。セーフティ・セキュリティを担保するための標準化活動は、日本としても大いに歓迎するべきものであるが、場合によっては、日本の主力産業にとっての非関税障壁にもなり得るものである。このようなことから、これらの国際規格が日本語にとって有益なものとなるよう積極的に標準化活動に参加すべき分野であると思われる。

また、これらの標準化に特化した SC/WG が存在するわけではない。SC7, SC22 以外にも IEC61508 にセキュリティ関連の規定が追加されるなど、複数の委員会を舞台に検討されているため、特定委員会の参加者に委ねるのではなく、包括的・継続的な情報収集や戦略検討が必要となると思われる。

第５章　全体のまとめ

ソフトウェアの属性情報管理の必要性は、認識されつつも出来る範囲に留まっており、それぞれの企業／団体の責任に委ねられているのが、現状である。

つながる世界の進展とともに、色々なソフトウェアが連携動作する環境においては、ソフトウェアの属性情報管理の必要性は更に増していくものと想定される。
流通するソフトウェアを一元的に管理しようという試みは、１つの企業／団体だけで成しうるものではなく、業界横断的な取り組みとして実現する必要がある。
本稿は、そのような取り組みの第１歩として、問題提起の位置づけと考えて頂きたい。

これを契機として、ソフトウェアの信頼性向上の取り組みの一環として、議論や取り組みが活発になることを期待したい。