# VRDA
# （Vulnerability Response Decision Assistance）

ハル・バーチ（Hal Burch）[1]、アート・マニオン（Art Manion）[1]、伊藤友里恵（Yurie Ito）[2]

[1] カーネギーメロン大学　ソフトウェア工学研究所　CERTコーディネーションセンター
アメリカ合衆国15213、ペンシルベニア州ピッツバーグ
{hburch,amanion}@cert.org

[2] 有限責任中間法人 JPCERT コーディネーションセンター
〒101-0054 東京都千代田区神田錦町3-17 廣瀬ビル11階
yito@jpcert.or.jp

**摘要：**毎年、何千もの新しいソフトウェア脆弱性が報告されており、影響を受ける組織はそれらを分析し、どのように対応するべきかを決定しなければならない。多くの組織では場当たり的に意思決定が行われており、その結果は多くの場合、組織全体としての考えが適切に反映されない一貫性のない決定になっている。VRDA（Vulnerability Response Decision Assistance：脆弱性対応意思決定支援システム）は、組織が他の組織の分析結果を利用したり、意思決定を体系化したりすることを可能にする。VRDAにより、組織は関心のない脆弱性の分析に投じる時間を減らし、決定の一貫性を向上させ、組織の目標に沿った意思決定を組み立てることが可能になる。VRDAは、データ交換フォーマット、意思決定モデル、意思決定モデル構築技法、および、これらを実現するためのツールで構成される。ある対応チームでは、VRDAの基本的な形態を採用して分析の対象となる脆弱性の数を半減させている。別の対応チームでは、その組織内でVRDAの実装の開発とテストを行っている。

**キーワード：**VRDA、KENGINE、脆弱性、意思決定支援、ディシジョンツリー、深刻度評価基準

## 1 概要

2006年にCERT/CCが記録した脆弱性の数は8,064件を超え [1]、NVDでは6,604件にのぼった [2]。組織は、個々の脆弱性を分析して、影響を受けるソフトウェアシステム、攻撃が成功した場合のインパクト、そして攻撃者が脆弱性の悪用に成功する難易度を判断する必要がある。組織はこの分析の完了後、該当する脆弱性について注意喚起を発したり、さらに詳しく分析したり、あるいは脆弱性に対して何らかの対応を行ったりするなど、さらに何らかの措置を講じるべきかを決定する必要がある。このようなコストのかかる分析が世界中の組織で繰り返し行われており、結果として作業の重複が大量に生じている。

分析が完了したら、組織は脆弱性に対してどのように対応すべきかを決定しなければならない。対応はさまざまである。問題を無視したり、パッチ適用プロセスを即座に開始したり、次回の定期更新のために問題を記録しておいたり、脆弱性に関する情報を（内部または外部に）公開したりするなどできる。特定の脆弱性に対する措置は、影響を受けるシステムの数、それらのシステムの価値、攻撃が行われる可能性の高さ、攻撃が成功した場合のインパクト、パッチのテストと適用にかかわるリスクとコストに基づく。

多くの組織では、どの措置を実施するかは、スタッフの経験に基づいて場当たり的に決定されている。結果として、組織の実際の目標ではなく、その決定を行った個人がその個人として認識している組織の目標を反映した、一貫性のない決定となる傾向がある。決定を行う人数を増やしても、問題は複雑になるだけである。

筆者は、VRDA（Vulnerability Response Decision Assistance : 脆弱性対応意思決定支援システム）という新しいシステムを提案する。VRDAは、意思決定プロセスを体系化することでこれらの問題に取り組む。この体系化により、分析の交換が可能になり、作業の重複が減る。また、組織の考え方をモデル化することで､決定に一貫性を持たせ､組織の目標との整合性を高められる。VRDAは、組織がどの脆弱性に対応しなければならないか、どのような対応をするべきか、そしてどのような優先順位で対応するべきかという問いに答えられるように作られている。

## 2 VRDA

VRDAは、FACT（事実関係。脆弱性の属性や特性）の詳細、影響を受けるシステムとFACT値との関係を記録する手段、データ交換の仕組み、データ交換フォーマット、意思決定のモデル化の形式、および意思決定モデルを作成するための方法で構成されている。組織がVRDAを最大限に活用できるように、VRDAはオープンかつモジュール型になっている。つまり、組織はそれぞれのニーズに最も合うコンポーネント（構成要素）を使うだけでよい。VRDAの各種コンポーネントについては、本文書の全体を通じて説明する。VRDAの使用法の概要を図1に示す。

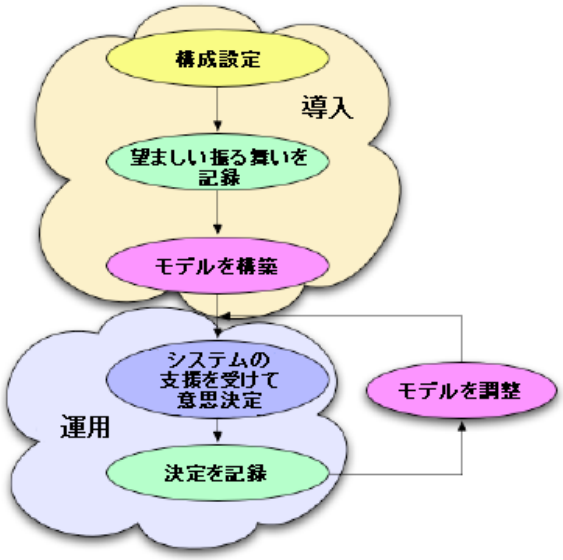


図1：VRDAシステムの概要

VRDAを使用する組織は導入時に、正確な対応の決定（タスクの実施）ができるように、どのような情報（FACT）が必要かを決めなければならない。VRDAは、選ばれたFACTとタスクで構成設定される。FACTは、上流の情報提供者から入手してもよい。つまり、脆弱性分析能力を有するCSIRT（Computer Security Incident Response Team：コンピュータセキュリティインシデント対応チーム）が提供するVRDA FACTのフィードを購読することができる。その後、組織のIT資産および業務運営を熟知している分析担当者がサンプルとなる一連の脆弱性報告に対する適切な対応（望ましい振る舞い）を記録してシステムを鍛えなければならない。このプロセスによって、組織の価値観が意思決定モデルに反映される。

意思決定モデルが実用に耐えうるものとなったら、VRDAを運用フェーズに移行できる。分析担当者が脆弱性のスコアを決めたり、上流のフィードに組織固有のデータを追加したりするのに応じて、VRDAによって適切な対応が提示される。必要に応じてモデルを調整できるように、実際の対応の決定も記録される。たとえば、VRDAによって常に特定の対応が提示されるが、分析担当者がそれに同意できないことが多い場合、そのモデルはおそらく組織の価値観を正確に反映していないか、VRDAが必要なFACTを使用して構成設定されていない可能性がある。

VRDAは、不完全な情報や分析担当者による「最善の推定」を許容するように設計されている。VRDAでは、決定を下すためにすべてのFACTが確認されるわけではなく、いくつかのスコアが少しずれていても、その決定に大きく影響する可能性は低い。モデルが適切に構築されていれば、組織に影響のない脆弱性報告を無視したり、深刻度の高い報告にフラグを設定したりするなど、すばやく正確な決定が簡単に下せるようになる。

## 2.1 FACT（事実関係）

VRDAにおいては、「FACT」は脆弱性に関する表明を意味する。FACTは、脆弱性の属性、性質または特性と考えることもできる。分析担当者がFACTを調べて記録する行為を「スコアリング」という。VRDAのFACTは、厳密な意味での「事実」ではない。つまり、VRDAのFACTは議論の余地がないということはなく、異なる解釈が存在しうるということである。FACTは、スコアリングの時点で分析担当者が入手できる最善の情報を表し、FACT値は部分的には分析担当者の判断や経験によって決まることもある。

VRDAでは、「中核」となるFACTのセットを提示する。しかし、組織は対応の決定に影響を与える任意のFACTを追加作成してスコアリングしてもよい。重要なFACTを考慮に入れないVRDAは、誤った決定を提示する可能性が高い。言い換えれば、VRDAによって使用されるFACTのセットは固定ではないということである。

ほとんどのFACTには、順序付けられた一定の範囲のとりうる値がある。正確であることの価値と、その正確性を実現するためのコストをバランスさせるために、FACT値の粒度はきわめて低い。たとえば、候補値が「low（低）」、「lowmedium（低－中）」、「medium-high（中－高）」および「high（高）」の4つというのは一般的である。VRDAでは、分析担当者が「無難」な値を選択する傾向を減らすために4つの値とするのが望ましい。自信を持って選択するために必要な情報が十分にない場合、分析担当者は知識や経験に基づいた直感的な推測を行うべきである。

VRDAのFACTは、Vulnerability（脆弱性）FACT、World（ワールド）FACT、Constituency（サービス対象）FACTの3つに分類される。これらの分類の違いは、意思決定もモデル化には影響しない。ただし、FACTがどのように該当し、それらをだれが提供するべきかを示すのには役立つ。

「Vulnerability FACT」は、脆弱性に直接かかわるFACTである。これらのFACTは、脆弱性固有の技術的な特性を示す。Vulnerability FACTは、ワールドFACTおよびConstituency FACTとは無関係に適用される。たとえば、サービス運用妨害（DoS）の脅威は、悪用コードの存在の有無や、該当する脆弱性を持つソフトウェアが特定のConstituencyに配備されているかどうかに関係なく存在する。Vulnerability FACTのスコアは誰でも決められるが、VRDAでは、経験豊かな分析担当者（CSIRTや対応チームなど）がVulnerabiity FACTのスコアを決め、それを下流のコンシューマに提供することが期待される。以下は、実用に耐えうるVulnerability FACTのリストである。詳細な説明は、VRDAの文書に記載されている。

セキュリティ製品 ― 脆弱性はセキュリティ製品に影響するか。（する／しない）

ネットワークインフラストラクチャ製品 ― 脆弱性はネットワークインフラストラクチャ製品に影響するか。（する／しない）

複数ベンダ ― 脆弱性は複数ベンダに影響するか。（する／しない）

影響1 ― 脆弱性によるシステムへの全体的な影響の程度。(低、低－中、中－高、高)

影響2 ― 脆弱性によるシステムの機密性、完全性、可用性への影響の程度。(低、低－中、中－高、高)

必要なアクセス経路 ― 脆弱性を悪用するために攻撃者が必要とするアクセス経路。(ルーティングあり、ルーティングなし、ローカルマシン、物理アクセス)

認証 ― 脆弱性を悪用するために攻撃者が必要とする認証レベルは何か。(なし、限定的、標準、特権)

必要なアクション ― 攻撃者が脆弱性を悪用するために必要となる、攻撃者以外の者によるアクションは何か。(なし、単純、複雑)

技術的難易度 ― 攻撃者が脆弱性を悪用しようとしたときに直面する技術的な難易度。(低、低－中、中－高、高)

「World FACT」は、脆弱性に関する「メタ」情報である。World FACTは、脆弱性が存在する世界、つまり環境の状態を表す。World FACTは、すべてのConstituency(サービス対象)にかかわるFACTである。World FACTのスコアは誰でも決められるが、VRDAでは、経験豊かな分析担当者がWorld FACTのスコアを決め、それを下流のコンシューマに提供することが期待される。以下は、実用に耐えうるWorld FACTのリストである。詳細な説明は、VRDAの文書に記載されている。

一般の関心 ― 脆弱性が一般から受けている注目の程度。(なし、低、低－中、中－高、高)

一般公開情報の質 ― 脆弱性について入手できる一般公開情報の質の程度。(許容不能、許容可能、高)

悪用活動 ― 存在する悪用または攻撃のレベル。(なし、悪用が存在、活動は不活発、活動は活発)

報告の情報源 ― 脆弱性を報告した人物または組織。

「Constituency FACT」は、脆弱性の情報を、特定のConstituency(サービス対象)に固有の尺度で測る。たとえば、CSIRTはどのように対応すべきかを決定するときに、サイト全体あるいはインターネット全体など、大きな範囲を考慮することが考えられる。システム管理者は、サイト、研究室、部署など、より狭い範囲を考慮することが考えられる。Constituencry FACTは、Constituencyの間で異なる可能性が高く、対応を決定するContituencry自身によって(または、そのConstituencyに代わって)提供されなければならない。

利用人口 ― Constituencyにおける脆弱性のあるシステムの利用人口。(なし、低、低－中、中－高、高)

利用人口にとっての重要性 ― Constituencyにおける脆弱性のあるシステムの重要性。（低、低－中、中－高、高）

DFS（Default Fact Sets：デフォルトFACTセット）は、FACTのセットについてあらかじめ設定されているFACT値である。DFSは、類似の脆弱性に対して一貫性のあるスコアを与えるのに役立つ。DFSは、影響のスコアリングを支援するために考案された。たとえば、攻撃者が任意のコードを実行できる脆弱性のほとんどは、機密性、完全性、可用性にかかわる「影響2」のFACT値が「高」であると分析担当者が判断したとする。この場合、この分析担当者は、脆弱性報告にDFSを適用したときに「影響2」のFACTを設定する「任意のコードを実行」というDFSを定義できる。このDFSは、デフォルトのFACTセットを適用することに加えて、その名前自体がFACTである（つまり、分析担当者が「任意のコードを実行」というDFSを適用したという事実を、対応の決定に織り込める）。その意味では、DFSはキーワードやタグに似ている。

### 2.2 LAPT（Lightweight Affected Product Tag）

脆弱性にかかわるFACTの交換を阻む障害の1つは、利用人口や利用人口にとっての重要性など一部の重要なFACTが、外部エンティティが決めることのできないConstituency FACTであるという点である。影響を受ける特定のシステムの利用人口が、あるConstituencyにおいて高かったとしても、そのConstituencyのサブセットにおいて利用人口が必ずしも高いとは限らない。たとえば、Microsoft Internet Explorerはインターネット全体では利用人口が高いが、組織によっては、ポリシとして別のブラウザを使用していたり、Microsoft Windowsシステムの数が少なかったりするなどの理由で、Microsoft Internet Explorerを使用しているシステムが少ない場合も考えられる。

結果として、Constituencry FACTは、一般にやり取りしてもあまり役立たない。LAPT（Lightweight Affected Product Tag：軽量被影響製品タグ）は、影響を受けるシステムのセットを伝えることで、Constituency FACTを計算できるようにする。脆弱性には、LAPTとして表現された、影響を受けるシステムのリストがタグ付けされる。VRDAは、LAPTを含む脆弱性情報を受け取ると、リストに挙げられている各LAPTについて、対応するConstituency FACT値をデータベースに照会する。そして、Constituency FACTごとに、LAPTの間で最も悪い（最終的に何らかの措置に結び付く可能性の高い）値に対応する値を選択する。たとえば、脆弱性が利用人口の高いLAPTと低いLAPTに影響する場合、利用人口は「高」に設定される。最も悪い値を選択する理由は、最善の推定値、また重要な脆弱性が隠れてしまう可能性が最も低い値を選ぶためである。脆弱性の中には、利用人口が低いシステムの大きな集合に影響するために、総合すれば利用人口が高くなる脆弱性がいくつかある。

このような組み合わせの場合、利用人口が低いけれども価値の高い製品と共に、利用人口が高いながらも価値の低い製品が影響を受けると、予想外の結果となることがある。この場合、結果としてConstituency FACTは、利用人口が高く、FACT値も「高」となるが、これは間違いであるといえるであろ

う。しかし根本問題は、LAPTではなく、FACTのモデル化における単純化にある。少数の高価値システムと多数の低価値システムを保有する利用人口は、利用人口とその人口にとっての価値という観点からモデル化するのは困難である。我々は、現在のモデルのほうが利便性が高いと信じているが、組織によっては、人口にとっての価値ごとに利用人口を構成するようにVRDAを設定できる。その場合には、この問題は生じない。

LAPTは、影響を受けるシステムの概要を記述するように考えられている。したがって、ベンダとシステムを示す。特定のバージョンの利用人口が大きく、多数の脆弱性がそのバージョンに影響を与えるのでない限り、バージョンは含まれない。たとえば、Apache Webサーバは単独のLAPTである。詳細を省く理由は次のとおりである。

1. バージョン情報の特定が困難。多くの場合、影響を受ける正確なバージョンを特定するが困難である。商用システムや、パッチを特定の順番で適用することを要求されないシステムの場合は特にそうである。
2. 目録維持のコスト。各ソフトウェア製品の規模について目録を維持するだけでも十分困難である。その上にバージョン番号も追跡するとなると、問題は何倍も難しくなる。
3. 利便性が限られる。発見される脆弱性のほとんどは、影響を受ける製品の現在のバージョン、および多くの場合、最近のすべてのバージョンに影響する。したがって、バージョン番号を調べることで切り捨てられる脆弱性の数は少ない。

ほとんどLAPTは、特定のソフトウェア製品を示す。脆弱性が特定の技術に影響する場合や、特定の技術の多数の実装に共通するエラーである場合、「技術LAPT」を利用することができる。たとえば、SSLや、多数のSSLライブラリに見られるエラーの場合に、影響を受けるすべての製品を挙げるのではなく、「SSL」の技術にかかわるものとすることができる。ただし、LAPTのリストに、影響を受けることが確認された製品も含まれていることもある。技術が一般的なものであれば、その技術を使用する製品のどのようなリストも不完全となる運命にある。技術LAPTは、こうした不完全性を回避するために、影響を受ける製品の分類を表す仕組みである。

### 2.3 データ交換

VRDAの目標の1つは、今日行われている脆弱性分析の重複を削減することにある。そのため、組織は構造化された脆弱性情報を他の組織からFACT値およびLAPTとして取得できる必要がある。情報を取得した組織は、必要に応じてFACT値を調整したり、他のFACT値を補ったりした後、その結果を他の組織で利用できるように再公開することができる。たとえば、国家レベルのCSIRTが企業のCSIRTに対して情報を公開し、企業のCSIRTは社内の各グループに情報を渡すことが考えられる。この第2レベルでは、注意を喚起したり、

脆弱性情報のVRDAフォーマットのサブセットに、場合によってはローカルの情報を追加したりするなどの方法がある。

組織は、脆弱性情報をFACT値およびLAPTとして受け取ったら、それまで知られていなかったLAPTの値があればそれを記録しなければならない。記録後、あるいは新しいLAPTがなければ、Constituency FACTを脆弱性情報に追加できる。この時点で、VRDAはその設定に基づいてフィルタ処理を行う。理想的には、大半の脆弱性が除外され、組織がそのリソースをより適切に集中できるようになる。このフィルタ処理で除外されなかった脆弱性には、組織がローカルのFACTについてFACT値を追加する。その後、「意思決定のモデル化」の項に示すプロセスに基づいて判断が提示される。

データ交換フォーマットは、VULDEF [3] に基づく。Solution、Related、ExploitなどVULDEFの大半の省略可能フィールドは、VRDAでは無視される。VULDEFに対してVRDA交換フォーマットの最も大きく異なる点は、AffectedItemにLAPTが追加され、FACTの値を伝えるためのFactListとFACTタグが追加されている点である。

交換フォーマットの一部省略された例を以下に示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<Vulinfo>
  <VulinfoID>JVN#178394</VulinfoID>
  <VulinfoData>
    <Affected version="1.0" historyno="2">
      <AffectedItem lapt="Microsoft-Windows-XP"/>
      <AffectedItem lapt="Tech-HTTP-Server"/>
    </Affected>
    <FactList version="1.0" historyno="2">
      <ImpactConfidentiality>Low</ImpactConfidentiality>
      <ImpactAvailability>Low</ImpactAvailability>
      <ImpactIntegrity>Medium</ImpactIntegrity>
      <AccessRequired>Routed</AccessRequired>
...
</VulinfoData>
</Vulinfo>
```

### 2.4 意思決定のモデル化

VRDAでは、プロセスをディシジョンツリーとしてモデル化することで意思決定を行う。ディシジョンツリーの例を図3に示す。ディシジョンツリーの評価はその根（ルート）から始まる。評価経路の途上にある各ノードには、属性に基づく子が関連付けられている。例として示したディシジョンツリーにおいては、利用人口が多い場合、左の子へ進み、そこで悪用の難易度が検討される。悪用の難易度が低いと、ディシジョンツリーの評価結果は「must（必須）」となる。

ディシジョンツリーを選択した理由は、その振る舞いが明確に示されており、手作業で修正が可能だからである。ディシジョンツリーは、過去の脆弱性に関する記録された意思決定に基づいて計算されることが期待されるが、

計算結果のツリーは調整が必要なことも考えられる。たとえば、組織がポリシとして、特定の情報源からの報告を個別に検証する必要がなかったり、特定の種類の報告に対しては必ず対応したりすることが考えられる。このようなポリシを反映するようにディシジョンツリーに修正を加えることが、明確かつ簡単にできる。ニューラルネットや線形結合など他の意思決定モデルのほうが、意思決定プロセスの複雑さを的確に捉えられるかもしれないが、それらによって実装されるポリシは把握するのが難しく、手作業による予測可能な方法でモデルを調整する手段が欠落している。

筆者は、結果として得られる決定は不完全であると想定している。しかし、それを補うために、ブール値の代わりに、段階のある判定を提示する。具体的には、「must（必須）」、「should（推奨）」、「might（検討）」、「won’t(不要)」の4レベルを使用する。目標は、結果として得られる決定が、「正しい」値と1レベルを超えて異ならないようにすることである。筆者の経験では、専門家の間でも意見がこれよりも大きく異なることがよくあるので、この正確さは野心的であるかもしれない。いずれにしても、この意思決定は、これを取り扱う者にとってはルールではなく、ガイドラインである。これにより、すべての決定について、評価結果が「対応不要」となった脆弱性を無視することを含め、優先順位付けの自動化が可能になり、組織内の担当者の負担が軽減される。

VRDAでは、2つの参考文献（[4]および[5]）に記述されている、エントロピーを最も削減する属性を再帰的に選択するアルゴリズムと似たアルゴリズムを使用してディシジョンツリーを構築する。ただし、ディシジョンツリーの構築後に枝刈りをするのではなく、構築時に、有意性のカイ二乗検定で有意と認められなかった属性を考慮しない。

ディシジョンツリーの構築後、組織はノードで使用している属性を変更し、アルゴリズムを使用してリーフ値あるいはサブツリー全体を計算して、ディシジョンツリーに変更を加えたり、リーフ値を変えたりできる。これらの操作によって、特定の種類の脆弱性の場合には必ず手作業でパッチを適用するなどの重要、かつ、よく理解されているポリシ決定を反映することが可能になり、その後、他の脆弱性のツリーを自動的に計算させられるようになる。

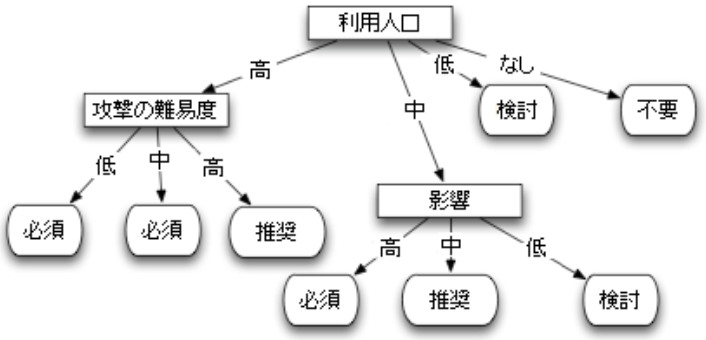


図3：ディシジョンツリーの例

## 3 現在の使用状況

CERT/CCでは、VRDAの限定的な実装を使用して、どの脆弱性報告が分析担当者の注目を必要とするかを判断している。簡単な調査の結果、2つのFACTがこの判断に大きな影響を与えることがわかった。すなわち、影響を受けるシステムの利用人口と、脆弱性のインパクトが及び範囲の広さである。この2つFACTを記録し、簡単な意思決定モデルを適用することで、分析担当者に割り当てられる報告の数は半減した。

JPCERT/CCは、KENGINEという名前のWebベースのVRDA実装を開発した。JPCERT/CCはKENGINEを内部で使用しており、いくつかのConstituentで試験運用を行っている。KENGINEは、VRDAの仕様を実装するほか、パフォーマンスと意思決定の振る舞いをモニタリングするワークフロー管理と報告の機能を備えている。JPCERT/CCは、脆弱性情報をVRDAフォーマットで発行し、KENGINEを一般に公開する予定である。

## 4 将来の方向性

VRDAを開発する各チームは、以下の実施を考えている。

1. 脆弱性報告のスコアを判定し、下流のコンシューマ、おそらくは一般にFACTを提供する。コンシューマは、これらのFACTを、意思決定支援コンポーネントと組み合わせるかどうかを問わず、自由に使用することができる。
2. コンシューマが意思決定支援コンポーネントを使用できるように文書とガイダンスを提供する。
3. 現実の環境における経験を得るために、ベータ版のVRDAシステムを有識者コンシューマに提供し、VRDAの有効性と可能性を判断する。
4. ベータ版の配備結果およびコミュニティからのフィードバックに応じてVRDAを調整する。
5. 他の脆弱性情報システム、とりわけCVSS（Common Vulnerability Scoring System）[6]との相互運用性を調べる。

## 5 関連研究

脆弱性情報を表現し、コンシューマが脅威度を判断して適切な対応を採れるようにする何らかの手段を提供する取り組みは、これまでいくつか行われてきた（または、現在行われている）。評価基準、情報交換フォーマットとプロトコル、および脆弱性情報データベースなどがこれらの取り組みに含まれる。VRDAは、これらの多くの取り組みの考え方や洞察を参考にしている一方で、脆弱性対応に対して意思決定支援のアプローチを提示する。

### 5.1 CVSS（Common Vulnerability Scoring System）

おそらく最も似ていると考えられ、かつ同時期の取り組みであるCVSS（Common Vulnerability Scoring System）は、「情報システムの脆弱性をランク付けし、その脆弱性の全体的な深刻度とリスクを表す複合的なスコアをエンドユーザに提示する」。[6] VRDAは、いくつかの面でCVSSに似ている。特に、脆弱性を記述するために使用するFACT（VRDA）と評価基準（CVSS）の表現が似ている。VRDAの意思決定支援コンポーネントをCVSSの評価基準と組み合わせて使用することは十分に可能だと考えられる。VRDAは、中核となるFACTを指定しているが、組織の対応の決定に大きく寄与するFACTであれば、考慮することは可能であり、考慮すべきである。これに対し、CVSSは固定の評価基準のリストを指定する。

これよりも大きな違いは、VRDAでは個々の対応の決定を生成するために意思決定支援の考え方を利用しているのに対し､CVSSでは評価基準に対して固定値を割り当て、深刻度の計算に単独の式を適用する点である。VRDAでは、組織が個別に独自の値を設定し、それぞれ独自に対応を決定できる。しかしこの設計には代償が伴う。つまり、VRDAはCVSSに比べて組織により多くの労力を要求する。また、公平を期して述べておくと、CVSSは個々の組織に固有の特性に基づいて総合スコアを修正する限定的な環境評価基準を備えている。

CVSSとVRDAは、脆弱性の特性を同じように評価するが、それぞれのシステムはいくらか異なる目標を持って設計されている｡CVSSは総合的な深刻度スコアを提示することを目的としているが、VRDAは個々の組織による脆弱性への対応方法にかかわる意思決定の側面を中心に据えている。

### 5.2 交換フォーマット

CAIF（Common Announcement Interchange Format）[7]、CMSI（Common Model of System Information）[8]、EISPP Common Advisory Format Description [9]、およびDAF（Deutsches Advisory Format）[10]を含め、多様な脆弱性情報交換フォーマットが存在する。これらのフォーマットは一般に、勧告文書で使用できる脆弱性情報を交換および提示する方法を示し、VRDAに不要な機能を含んでいる。中には、VRDAの要件に適合するように合理的に拡張が可能でないフォーマットもある。

### 5.3 その他の研究

その他の関連する研究としては、次のような研究がある（順不同）：Purdue UniversityのCERIAS Cassandraツール[11]、CERT Coordination Center/US-CERT Metric [12]、MITRE のCVE（Common Vulnerabilities and Exposures）[13]とOVAL （ Open Vulnerability and Assessment Language ） [14] 、 VULDEF（VULnerability Data publication and Exchange Format）[3]（VRDA交換フォーマットの基礎として使用した）、NIST ICAT（現在は廃止）[15]、NVD（National

Vulnerability Database。ICATの後継）[2]、VuXML（Vulnerability and eXposure Markup Language）[16]、OSVDB（Open Source Vulnerability Database）[17]、SIGVI [18]。

## 参考文献


1. CERT/CC Statistics 1988 – 2006, http://www.cert.org/stats/
2. National Vulnerability Database (NVD) Statistics, http://nvd.nist.gov/statistics.cfm
3. Terada, M.: VULDEF: The VULnerability Data publication and Exchange Format data model, http://jvnrss.ise.chuo-u.ac.jp/jtg/vuldef/index.en.html
4. Russell, S., Norvig P.: Artificial Intelligence: A Modern Approach. Prentice-Hall (1995)
5. Moore, A.: Decision Trees, http://www.autonlab.org/tutorials/dtree.html
6. Forum of Incident Response Teams: Common Vulnerability Scoring System (CVSS), http://www.first.org/cvss/, http://www.first.org/cvss/cvss-guide.html
7. RUS-CERT: Common Announcement Interchange Format (CAIF), http://www.caif.info/
8. Grobauer, Bernd: CVE, CME,..., CMSI? – Standardizing System Information, http://www.first.org/conference/2005/papers/dr.-bernd-grobauer-paper-1.pdf
9. European Information Security Promotion Programme (EISPP): Common Advisory Format Description 2.0, http://www.eispp.org/commonformat_2_0.pdf
10. Deutscher CERT-Verbund: Deutsches Advisory Format (DAF), http://www.certverbund.de/daf/index.html, 2004.
11. CERIAS Cassandra tool, https://cassandra.cerias.purdue.edu/main/index.html
12. US-CERT Vulnerability Notes Field Descriptions – Metric, http://www.kb.cert.org/vuls/html/fieldhelp#metric
13. Common Vulnerabilities and Exposures (CVE), http://cve.mitre.org/
14. Vulnerability and Assessment Language (OVAL), http://oval.mitre.org/
15. ICAT Metabase, http://icat.nist.gov/icat_documentation.htm, http://web.archive.org/web/20050320143644/http://icat.nist.gov/icat_documentation.htm
16. Vulnerability and eXposure Markup Language (VuXML), http://www.vuxml.org/
17. OSVDB: The Open Source Vulnerability Database, http://osvdb.org/
18. SIGVI, http://sigvi.sourceforge.net/what_is.php