脆弱性がもたらす影響をトレース可能な遷移グラフの提案

神宮 真人 †　　グレゴリー ブラン †　　奥田 剛 †　　山口 英 †

† 奈良先端科学技術大学院大学情報科学研究科
630-0192 奈良県生駒市高山町 8916 番地の 5
{masato-j, gregory, okuda, suguru}@is.naist.jp

あらまし 既知の脆弱性を悪用した不正アクセスは，提供された修正パッチを適用することで未然に防ぐことが出来る．しかし，自身が受ける影響を理解出来ないユーザは脆弱性に対しての危険性を認知出来ないため対策を行わない．本研究では，脆弱性とユーザが受ける影響の隔たりを除くことを目的とする．脆弱性がシステムにもたらす可能性のある影響を提示することで，ユーザに危険性を認知させることが出来ると推測できる．そこで我々は，脆弱性がもたらす影響をトレース可能な遷移グラフを提案する．さらにシステム情報をスキャンし，システムが受ける可能性のある攻撃パターンを特定する試作システムを開発した．

A Transition Graph to Trace the Vulnerabilities up to its Effects

Masato Jingu†　　Gregory Blanc †　　Takeshi Okuda †　　Suguru Yamaguchi†

†Graduate School of Information Scinece, Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-city, Nara 630-0192 JAPAN
{masato-j, gregory, okuda, suguru}@is.naist.jp

Abstract Illegal access that utilizes known vulnerabilities can be prevented beforehand by applying the patch provided by the vendor. But, users who can't understand how they are impacted won't apply countermeasures because they are not aware of the risk brought by the vulnerabilities. In this paper, the goal is to correlate the vulnerabilities with the impact the user receives. We expect that the user can acknowledge the risk by representing the potential impact the vulnerabilities bring to the system. Then, we propose a transition graph which can trace such impact. In addition, we developed a prototype able to scan the vulnerabilities present in the system and identify the attack patterns that may strike the system.

1　はじめに

不正アクセスは，システムの脆弱性を悪用することでおこなわれる．既知の脆弱性を悪用した不正アクセスについては，ベンダーが提供する修正パッチを適用することで未然に防ぐことができる．しかし，既知の脆弱性を利用した不正アクセスによる被害が後を絶たないことから，脆弱性が公開されても対策を行わないユーザが多いことがわかっている．[1]

不正アクセスによってシステムを中継リソースとして悪用されることによって，脆弱なシステムのユーザだけでなく，ネットワーク上の他のシステムにも被害が及ぶ．そのため，セキュリティ対策を行わないユーザを減らすことは安全なインターネットを構築するための一つの課題となっている．

脆弱性対策を行わないユーザは，二種類の

ケースが考えられる．一つのケースは，セキュリティに対する意識はあるが，ソフトウェアのアップデートに伴い発生するシステムの可用性の低下や，システムに発生する不具合を懸念し，対策を行えないケースである．もう一つのケースは，セキュリティに対する意識が低く，脆弱性を放置することにより発生する可能性のある影響を理解していないケースである．

本研究では，後者のケースを対象にしている．脆弱性情報は専門用語が多く含まれており，専門知識を有さないユーザにとっては自身への影響を理解することが困難である．そこで，脆弱性情報とシステムが受ける可能性のある影響を関連付けることに着目した．

システムに内在する脆弱性をスキャンし，内在する脆弱性によってシステムが受ける可能性のある影響をユーザに提示するシステムを設計した．また試作システムとして，システムにインストールされたソフトウェアの情報をスキャンし，システムが受ける可能性のある攻撃パターンを特定するシステムを開発したため，これを報告する．

以下，第 2 章では関連研究について述べる．第 3 章では，提案するシステムについて述べる．第 4 章では，試作したシステムと実行結果について述べる．第 5 章では，提案システムで検討すべき課題について述べる．最後に，第 6 章で本研究のまとめを述べる．

2　関連研究

これまでに，脆弱性の影響度を分析する研究がいくつかおこなわれている．文献 [2] では，ネットワークシステムの表現モデルである NSQ モデル [3] を用いて，ネットワークシステムにおける脆弱性の影響度を定量化した．

原田らは，FIRST が推進する共通脆弱性評価システムである CVSS[4] を用いて，ネットワークシステムにおける脆弱性影響度範囲の検討をおこない，冗長化効果および中継機器における可用性影響の大きさを測定した．これらの研究は，ネットワークシステムにおける脆弱性の波及範囲を定量化している．本研究では，端末ベースでの影響度を特定することで，単一システム内で発生する影響を特定することを目的とする．

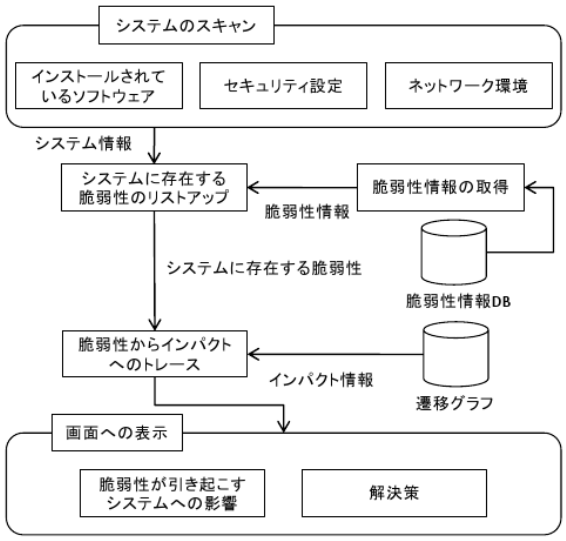

図 1: システムの概要

3　遷移グラフ提示システムの提案

本章では提案するシステムについて詳述する．

3.1　要件定義

本研究ではユーザが自身のシステムに起こり得る影響を理解することを目的とするため，提案システムでは下記の要件を定義する．

- システムに内在する脆弱性が抽出できる
- 脆弱性とシステムへの影響が関連付けられる

3.2　提案システムの概要

本節では，我々が提案するシステムの概要を述べる．本システムは，システムに内在する脆弱性をスキャンし，システムが受ける可能性のある影響を提示することが可能なシステムである．システムの概要を図 1 に示す．

システムに内在する脆弱性は，インストールされているソフトウェアとセキュリティ設定に

起因する．そのため，システムのスキャン部では，インストール済みソフトウェアの情報とセキュリティ設定を取得する．

システム情報は OpenSCAP[5] を用いて取得する．OpenSCAP は，セキュリティチェックリストを記述するための仕様である XCCDF[6] で記述された設定項目を，チェック方法について記述するための仕様である OVAL[7] を利用してセキュリティ問題をチェックするプログラムである．

次に，リストアップされたソフトウェアの脆弱性情報を取得する．脆弱性情報には，NIST が管理する脆弱性情報データベースである NVD[8] を利用する．NVD は，MITRE 社が管理するベンダ非依存な共通脆弱性識別子である CVE[9] で命名された脆弱性情報を扱っている．CVE では，CVE 識別番号によって脆弱性が一意に識別されている．NVD では，識別された脆弱性の種類や重要性，脆弱性が発見されたソフトウェア名とバージョン，ソフトウェアを開発した企業名などが登録されており，毎日更新されている．このデータは一般に公開されており，XML 形式で取得が可能である．

これらの脆弱性情報から，後述する遷移グラフを作成する．画面への表示部では，遷移グラフによってトレースされた結果と解決策をユーザに提示する．

3.3 遷移グラフ

遷移グラフとは，脆弱性と，脆弱性が持つ特性，及びシステムへ与える可能性のある影響を関連付けたグラフである．遷移グラフを用いることで，システムをスキャンした結果から，システムへ与える可能性のある影響を特定することが可能となる．遷移グラフのイメージを図 2 に示す．

遷移グラフは，脆弱性情報と脆弱性タイプ情報，攻撃パターン情報で成り立っている．それぞれの情報が関連付けられており，脆弱性から脆弱性タイプと攻撃パターンをトレースすることが可能となる．図中の各ノードは，識別番号の他にユーザに提示すべき情報を持っている．

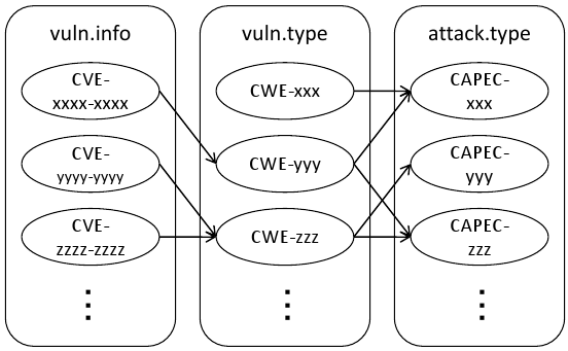

図 2: 遷移グラフのイメージ

脆弱性情報には，NVD を利用する．NVD からは下記の情報を取得する．

- vuln:cve-id (CVE 識別番号)
- vuln:cwe-id (CWE 識別番号)
- vuln:vulnerable-software-list (ソフトウェア名とバージョン)

脆弱性に伴う影響の原因や対策を一意に識別するために，脆弱性が持つ特性をカテゴライズする必要がある．脆弱性のカテゴライズには，MITRE 社が中心となり仕様策定が行われた共通脆弱性タイプ一覧である CWE[10] を利用する．CWE は，SQL インジェクション，クロスサイト・スクリプティング，バッファオーバーフローなど，多種多様にわたるソフトウェアの脆弱性を脆弱性タイプとして分類している．また，それぞれに CWE 識別子を付与し，階層構造で体系化している．CWE からは，脆弱性タイプ情報として下記の情報を取得する．

- Name (脆弱性タイプ名)
- Description_Summary (概要)
- Common_Consequences (一般的な影響)
- Detection_Methods (被害の暖和策)
- Relationship (関係性)
- Affected_Resource (影響を受けるリソース)
- Related_Attack_Patterns (関連する攻撃パターン)

脆弱性の存在によって被害を受ける可能性のある攻撃手法を識別するために，攻撃手法に一意の識別子が必要となる．攻撃手法の識別には，MITRE 社が推進する攻撃パターン一覧である CAPEC[11] を利用する．また，それぞれに CWE 識別子を付与し，階層構造で体系化している．CAPEC からは，攻撃パターン情報として下記の情報を取得する．

- Summary (概要)
- Solutions_and_Mitigations (被害の暖和策)
- Payload_Activation_Impact (ペイロードが実行されることにより受ける影響)
- CIA_Impact (機密性，保全性，可用性に与える影響)

4 試作システムの実装

本章では，3 章で述べた設計を基に実装した試作システムと，試作システムの実行結果について詳述する．

4.1 試作システム

本節では，実装した試作システムについて詳述する．試作システムは脆弱なシステム環境で稼働させる必要があるため，脆弱なソフトウェアをインストールした FreeBSD8.0 を構築し，その上で実装した．試作システムは Perl で作成し，XML パーサモジュールとして Perl の XML::Simple を利用した．

試作システムは下記の機能を実装している．

機能 1 システムにインストールされているソフトウェアを抽出する

機能 2 NVD からソフトウェアに含まれる脆弱性を抽出する

機能 3 CWE 及び CAPEC から脆弱性の特性とシステムが受ける可能性のある攻撃パターンを抽出する

Algorithm 1 システム情報から攻撃パターン情報へのトレース

```
SoftList ← パッケージ管理ソフトのリスト
nvd ← NVD の XML データ
for all entry in nvd do
  if SoftList is < vuln : vulnerable−software−list > then
    cve ←< vuln : cve − id >
    cwe ←< vuln : cwe − id >
  end if
end for
cwe ← CWE の XML データ
for all entry in cwe do
  if cwe is < CWE − ID > then
    capec ←< CAPEC − ID >
  end if
end for
capec ←CAPEC の XML データ
for all entry in capec do
  if capec is < CAPEC − ID > then
    return entry
  end if
end for
```

機能 1 では，システムにインストールされているソフトウェアをスキャンするために，パッケージ管理ソフトを用いた．パッケージ管理ソフトでインストールされたソフトウェアに限り，ソフトウェア名，バージョン情報を取得する事ができる．機能 2 では，スキャンしたソフトウェアに対して脆弱性の有無を確認するために，XML で記述された NVD から，ソフトウェア名とバージョンを正規表現を利用して特定する．機能 3 では，XML で記述された CWE の情報から，該当する CVE 識別番号の脆弱性タイプを特定する．また，XML で記述された CAPEC の情報から，該当する脆弱性タイプが受ける攻撃パターンを特定する．

次に試作システムの擬似コードを Algorithm1 に示す．まず，パッケージ管理ソフトを用いてシステムにインストールされているソフトウェアのソフトウェア名，バージョンを抽出する．次に，NVD から取得した XML データに含まれるタグ vuln:vulnerable-software-list の要素に，システムにインストールされていたソフトウェアが含まれているか正規表現を用いて確認する．vuln:vulnerable-software-list の要素と，システムにインストールされているソフトウェア名とバージョンが一致した場合は，該当エントリの vuln:cve-id と vuln:cwe-id の要素を抽出する．そして，vuln:cwe-id の要素をキーに，CWE か

ら取得した XML データのタグ CAPEC_ID の要素を抽出する．最後に，CAPEC_ID の要素をキーに，CAPEC から取得した XML データに含まれる情報を抽出する．

4.2 試作システムの実行結果

本節では，CVE 識別番号：CVE-2010-1205 の脆弱性を持つシステムにおいて試作システムを実行した結果の一部を示す．CVE-2010-1205 は，PNG ファイルに関するライブラリである libpng に発見された問題であり，任意のコードの実行を許すバッファオーバーフローの脆弱性である．

出力された CWE の情報より，この脆弱性はメモリバッファ上でオペレーションを行うソフトウェアにおいて，メモリ内に意図するバッファの境界外へ読み書きが可能な際に発生するタイプの脆弱性であることが分かる．また，出力された CAPEC の情報より，下記の攻撃パターンを受ける可能性があることが分かる．

- 環境変数を通したバッファオーバーフロー (図 3, 5 行目)
- バッファオーバーフロー (図 3, 6 行目)
- クライアントサイドのインジェクションで誘発されるバッファオーバーフロー (図 3, 7 行目)
- バッファオーバーフローによるフィルタの故障 (図 3, 8 行目)
- MIME 変換 (図 3, 9 行目)
- バイナリリソースファイルのバッファオーバーフロー (図 3, 10 行目)
- シンボリックリンクを通したバッファオーバーフロー (図 3, 11 行目)
- 変数とタグのバッファオーバーフロー (図 3, 12 行目)
- パラメータ展開を通したバッファオーバーフロー (図 3, 13 行目)

1	Affected Software:png-1.2.32
2	CVE:CVE-2010-1205
3	CWE:CWE-119
4	Name:Improper Restriction of Operations within the Bounds of a Memory Buffer
5	CAPEC:10:Buffer Overflow via Environment Variables
6	CAPEC:100:Overflow Buffers
7	CAPEC:14:Client-side Injection-induced Buffer Overflow
8	CAPEC:24:Filter Failure throug Buffer Overflow
9	CAPEC:42:MIME Conversion
10	CAPEC:44:Overflow Binary Resource File
11	CAPEC:45:Buffer Overflow via Symbolic Links
12	CAPEC:46:Overflow Variables and Tags
13	CAPEC:47:Buffer Overflow via Parameter Expansion
14	CAPEC:8:Buffer Overflow in an API Call
15	CAPEC:9:Buffer Overflow in Local Command-Line Utilities

図 3: 実行結果

- API 呼び出しにおけるバッファオーバーフロー (図 3, 14 行目)
- ローカルのコマンドライン・ユーティリティにおけるバッファオーバーフロー (図 3, 15 行目)

この結果より，試作システムによって，システムに潜む脆弱性と，脆弱性の特性，及び受ける可能性のある攻撃パターンを導いた．

5 ディスカッション

今後検討すべき課題の一つは，システムのユーザへの情報の提示方法である．試作システムから取得した情報を羅列するだけでは，専門知識を持たないユーザが自身への影響を理解するには不十分である．コンピュータアーキテクチャを理解していないユーザに技術的な危険性を認知させるためには，技術的な攻撃のストラクチャを身近な事象に比喩して表現する方法が考えられる．

また，複数の脆弱性の存在によって起こり得る影響に関して検討する必要がある．単一の脆弱性によって成立する攻撃が僅かな被害しか与えないものであっても，特定の特徴を持った複数の脆弱性が同一システム内に存在した場合，大きな被害を受ける攻撃が成立する可能性がある．例として，権限昇格の脆弱性とコマンドインジェクションの脆弱性が内在する場合，より

権限のあるユーザでシステムに対してインジェクションをおこなう事が可能となる．よって，単一の脆弱性に対しての影響だけでなく，システムに内在するすべての脆弱性との関連性を視野に入れる必要があると考えられる．

6 まとめ

本研究では，システムに内在する脆弱性から，脆弱性がもたらす影響をトレース可能な遷移グラフを提案した．また，システムにインストールされているソフトウェアの情報からシステムが受ける可能性のある攻撃パターンを特定する試作システムを開発した．試作システムでは，NVD が提供する脆弱性情報と，CWE や CAPEC によって策定されている仕様に基づいたデータを用いた．その結果，システムに潜む脆弱性と，脆弱性の持つ特性，及びシステムが受ける可能性のある攻撃パターンを特定することができた．

今後の課題として，提案手法の評価が考えられる．自身が受ける可能性のある影響を認知することが，どの程度対策へのモチベーションとなるのか定量的に評価する必要があると考えられる．

参考文献

[1] サイバークリーンセンターボット対策プロジェクト．平成 20 年度サイバークリーンセンター活動報告. https://www.ccc.go.jp/report/h20ccc_report.pdf.

[2] T. Harada, A. Kanaoka, E. Okamoto, M. Kato. Identifying Potentially-Impacted Area using CVSS for Networked Systems. *Proceedings of The First Workshop on Convergence Security and Privacy (CSnP)*, July 2010.

[3] 金岡晃, 藤堂伸勝, 加藤雅彦, 岡本栄司．ネットワークシステムの安全性定量化に向けた新たな表現モデルとアクセス制御解析. 2008 年 暗号と情報セキュリティシンポジウム (SCIS), 2008.

[4] National Institute of Standards and Technology. National Vulnerability Database CVSS Scoring. http://nvd.nist.gov/cvss.cfm.

[5] OpenSCAP. http://www.open-scap.org/.

[6] National Institute of Standards and Technology. XCCDF. http://scap.nist.gov/specifications/xccdf/.

[7] MITRE Corporation. OVAL. http://oval.mitre.org/.

[8] National Institute of Standards and Technology. National Vulnerability Database Home. http://nvd.nist.gov/.

[9] MITRE Corporation. CVE. http://cve.mitre.org/.

[10] MITRE Corporation. CWE. http://cwe.mitre.org/.

[11] MITRE Corporation. CAPEC. http://capec.mitre.org/.