

脆弱性情報を用いたセキュリティ保護システムの試験導入と評価

衣川 達¹⁾, 竹原 一駿¹⁾, 中村 友昭²⁾, 大野 真伯¹⁾, 山下 俊昭¹⁾,
宗雪 勝也¹⁾, 小野 滋己¹⁾, 喜田 弘司¹⁾, 後藤田 中¹⁾, 最所 圭三¹⁾

1) 香川大学

2) 無所属

s23g355@kagawa-u.ac.jp

Testing and Evaluation of Security Protection Systems Using Vulnerability Information

Itaru Kinugawa¹⁾, Ichitoshi Takehara¹⁾, Tomoaki Nakamura²⁾, Masanori Ono¹⁾,
Toshiaki Yamasita¹⁾, Katsuya Muneyuki¹⁾, Shigemi Ono¹⁾, Koji Kida¹⁾,
Naka Gotoda¹⁾, Keizo Saisho¹⁾

1) Kagawa Univ.

概要

近年、組織に対する脆弱性を利用した攻撃が増加している。本研究では、組織内の機器情報を収集し、公開されている脆弱性情報をマッチングすることで、組織内の脆弱性を持つ機器を特定するセキュリティ対策システムを開発した。開発したシステムを香川大学の情報メディアセンターで被験者に対し、試験導入を行った結果検出精度が不十分であることが分かった。また、評価の際、UIがなく、システムの運用に手間がかかることが分かった。本稿では、精度改善のため、情報を修正する補助機能、運用のためのUIを開発し、試験導入と評価結果について述べる。

1 はじめに

近年、脆弱性によるサイバー攻撃の被害が拡大している。IPAが公開している「情報セキュリティ10大脅威」2024(組織編) [1]によると5位と7位に脆弱性を利用した攻撃についての記述がある。さらに、IBM Security XForceの脅威インテリジェンス・インデックスによると、教育機関に対する攻撃の割合は増加している [2]。しかし、大学等の組織においては、情報を手動で管理し、古い機器情報のまま放置されていること、脆弱性の情報が入ると資料を目視で比べて判断を行うこともある。このような状況から機器の管理が正確に行われていないことが多く、脆弱性が放置されたままになっている。よって、脆弱性対策は急務といえる。

我々はこの問題を解決するため、組織内の機器を一元管理し、公開されている脆弱性情報を元に組織内の機器が持つ脆弱性を検知するセキュリティ対策システム“BEYOND”を開発する [3]。以前は、機器情報と脆弱性情報の収集、脆弱性の検知を行い、半分は検知できている状態だった。

本稿では、機器情報の修正機能を追加し、検知率改

善の検証と、運用を開始するためにUIを開発し、試験導入を行った結果について述べる。

2 BEYOND

2.1 BEYOND 概要

BEYOND全体の構成を図1に示す。脆弱性情報収集部、IT資産管理部、影響算出部、ネットワーク制御部、更新支援部、フロント部で構成される。

脆弱性情報収集部にて、インターネット上に公開されている脆弱性情報を収集しデータベース(以下、DB)化する。IT資産管理部にて、組織内の機器情報をエージェントソフトウェア及びUIからの入力によって取得する。機器情報にはソフトウェア情報(ソフトウェア名とバージョン番号)、サービス情報(サービスの停止可能期間やポート番号)が含まれる。影響算出部にて、収集した脆弱性情報と機器情報を突き合わせ、脆弱性を検知する。検知した脆弱性の深刻度と機器の利用状況から対策方針を算出し、機器の利用者や情報システムの管理者に通知を行う。影響度の高い脆弱性が発見された場合は、ネットワーク制御部にて、ネットワークから遮断、隔離を行い対処する。更新支援部では、ソフトウェアの更新を自動で行うスクリプトを配

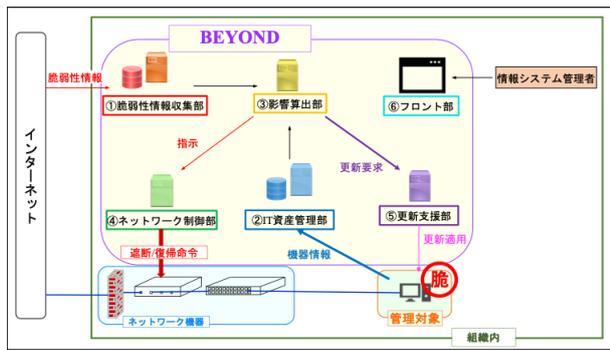


図1 BEYOND 全体図

表1 PC ルームの管理システムの結果 [3]

	BEYOND で検知可能	BEYOND で検知不可能
脆弱性有り	2 件	2 件
脆弱性無し	5 件	19 件

表2 インストーラーのダウンロードサイトの結果 [3]

	BEYOND で検知可能	BEYOND で検知不可能
脆弱性有り	3 件	3 件
脆弱性無し	8 件	37 件

表3 評価実験の結果 [3]

対象機器	適合率	再現率	F1 値
PC ルームの管理システム	0.5	0.28	0.36
インストーラーのダウンロードサイト	0.5	0.18	0.35

布することで脆弱性の解消を行う。フロント部にて、これらの機能を UI で制御する。本稿ではフロント部について述べる。

2.2 前回の実験の評価

前回の実験について述べる [3]. 前回は香川大学の PC ルームの空き情報管理システムとインストーラーのダウンロードシステムを対象に行った。評価実験の結果を表 1 と表 2 に示す。

この結果から適合率、再現率、F 値を求めると、それぞれ以下の表 3 に示す結果となった。

2.3 BEYOND 本格運用のための課題

評価より、脆弱性検知における再現率が低い結果を得た。これは、ソフトウェア名とバージョン情報にマッチングできない形式が存在している原因であると考えた。また、本評価を通じて、情報の登録や修正にコマンドラインを使う必要があるため、非常に手間がかかることが分かった。ここで、下記の課題を解決することで本格運用を可能とする。

1. マッチングできない形式を手動で修正した後の継続的な修正内容の反映。

2. コマンドラインを必要としない各部を統括する WebUI の開発。

以上の課題を解決するため、各機構の制御を行う UI を実装、脆弱性検知の再現率を向上させるためのソフトウェア情報修正機能を開発した。

3 運用のための開発

3.1 ソフトウェア情報修正機能

中村により実装されている [4]. 図 2 のようなパターンに対応出来ておらず、今後も未対応のパターンが収集されることになると考えた。そこで、未対応のパターンをシステム管理者が修正し、その修正内容を自動で継続的に適用する修正機能を開発した。

修正は図 3 の流れで行われる。手順は以下の通りである。

- 1 回目 導入したエージェントからソフトウェア情報が IT 資産管理部に送信される。送信されたソフトウェア情報は修正機能で修正履歴と比較される。修正履歴に無い場合、未対応のパターンに対し、当修正機能を用いて WebUI からシステム管理者が修正作業を行う。
- 2 回目以降 導入したエージェントから 1 回目と同様のソフトウェア情報が IT 資産管理部に送信されたとき、修正機能が修正履歴を参照し、同様のパターンがあれば修正履歴に従って自動で修正を行う。

修正に使用する UI を図 4 に示す。ソフトウェア情報の一覧から未対応のソフトウェア情報を選択すると、修正フォームが表示される。現在のソフトウェア情報が表示されるため、この情報を参考にしながら利用者は適切な修正を行う。入力後、送信ボタンを押すと情報が DB に反映される。

修正作業の負担について述べる。図 2 の表 1, 2 行目のパターンは、パッケージソフトウェアやレジストリ空の収集で見られる。ここで収集されるソフトウェア情報は標準でインストールされている物が大半であるため、手動で修正する件数は少数であると考えられる。図 2 の表 3 行目のパターンは、修正を繰り返すことで入力時のミスを発見し、徐々に減少していくと考えられる。

3.2 UI 設計

本研究で開発する WebUI の遷移図を図 5 に示す。WebUI の流れについて説明する。利用者はログイン

ソフトウェア名にバージョン番号が含まれる	修正前	ソフトウェア名	バージョン番号
		OpenSSL 1.1.1d (64-bit)	Null
		Microsoft SQL Server 2017_T-SQL [X][X][X][X][X][X][X]	14.0.1000.169
		Apache	2.4.37

文字化けした情報が登録される

WebUIからの入力時におけるスペルミス

図2 未対応パターン [3]

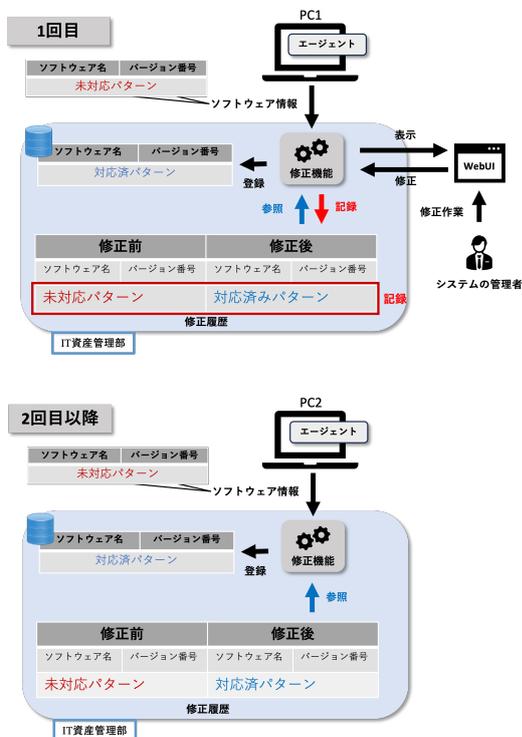


図3 自動修正機能 [3]

画面にアクセスし、ログインを行う。ログインに成功するとホーム画面に遷移する。ホーム画面では、機器情報や脆弱性情報の検索や閲覧を行う画面などのリンクが表示されている。利用者は、機器情報一覧画面から機器情報の登録を行う画面に移動し、機器情報を登録する。その後、機器にインストールされているソフトウェア情報登録画面に移動し、ソフトウェア情報を登録する。もし、機器がサービスを提供している場合、サービス情報登録画面に移動し、サービス情報を登録する。機器情報を登録した後、脆弱性検知画面にて、機器に存在する脆弱性を検知する。

次にソフトウェア情報の修正を行う場合、ソフトウェア情報修正画面に移動し、ソフトウェア情報を修正する。

最後に、脆弱性情報の検索を行う場合、脆弱性情報検索画面に移動し、検索条件を入力し、検索を行う。ここでは、検索条件に CVE - ID またはソフトウェ

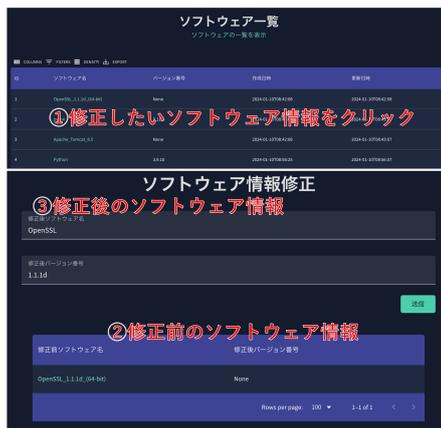


図4 ソフトウェア情報修正 UI [3]

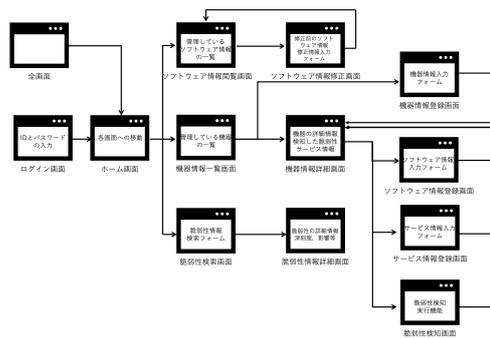


図5 UI 遷移図

ア名を入力することができる。ソフトウェア名を入力した場合、該当するソフトウェアに関連する脆弱性の CVE - ID の一覧が表示される。CVE - ID を入力した場合、脆弱性情報詳細画面に移動し、脆弱性の詳細情報を閲覧できる。全ての画面から、ホーム画面に戻ることが出来る。

4 評価

4.1 ソフトウェア情報修正機能の性能評価

本節では、ソフトウェア情報修正機能を用いることで脆弱性検知の再現率が向上するか検証する。

4.1.1 評価対象

評価対象には、情報メディアセンターで運用している基盤サーバの情報を用いる。評価に用いるサーバの情報を表4に示す。評価実験に使用する脆弱性情報は、JVN iPedia は1998年から2024年1月31日まで、NVD は2002年から2024年1月31日までに収集したものを使用する。

4.1.2 評価方法

4.1.1節で示したサーバを用いて、ソフトウェア情報修正機能を用いた場合と用いない場合で脆弱性検知の再現率を比較する。まず、LDAPサーバにインストー

表 4 評価対象のサーバ情報

サーバ名	OS	ソフトウェア情報の件数
LDAP サーバ	RedHat Enterprise Linux 7.4	1376 件
SMTP サーバ	RedHat Enterprise Linux 7.4	1366 件

表 5 脆弱性検知の区分分け

	BEYOND で検知可能	BEYOND で検知不可能
脆弱性有り	正常検知 (TP)	検知漏れ (FN)
脆弱性無し	誤検知 (FP)	TN

表 6 LDAP サーバの修正前の混同行列

	BEYOND で検知可能	BEYOND で検知不可能
脆弱性有り	94 件	75 件
脆弱性無し	6 件	1201 件

ソフトウェア名	バージョン番号
@anaconda	PackageKit-gtk3-module.x86_64
1.1.5-1.el7	PackageKit-yum.x86_64

追加リポジトリの名前

本来のバージョン番号

図 6 未対応パターンの例

ルされているソフトウェアに対して脆弱性の検知を行う。検知を行った結果と手動で正誤判定を行った結果を比較し、混同行列を作成する (表 5)。検知可能であり、脆弱性のある物を正常検知、検知不可能だったものを検知漏れ、検知した中に脆弱性が無かったものを誤検知と定義する。次に、ソフトウェア情報修正機能を用いて、未対応パターンのソフトウェア情報を修正する。その後、再度脆弱性の検知を行い、混同行列を作成する。最後に、SMTP サーバのソフトウェア情報を登録する。この時、LDAP サーバの修正履歴を基にソフトウェア情報修正機能が修正を行う。修正されたソフトウェア情報に対して、脆弱性の検知を行い結果を比較する。

4.1.3 評価結果

まず、LDAP サーバに対して脆弱性の検知を行った結果を表 6 に示す。ここで、未対応パターンのソフトウェア情報を図 6 に示す。未対応パターンのソフトウェア情報は、BEYOND で検知不可能なソフトウェア情報の 75 件と 1201 件の内 569 件存在した。

次に、ソフトウェア情報修正機能を用いて、LDAP サーバの未対応パターン 569 件を修正した後の結果を表 7 に示す。未対応パターンを修正した事により、75 件の検知漏れの内、74 件が BEYOND によって追加で検知することができた。追加で検知した脆弱性の

表 7 LDAP サーバの修正後の混同行列

	BEYOND で検知可能	BEYOND で検知不可能
脆弱性有り	166 件	1 件
脆弱性無し	8 件	1201 件

74 件の内、72 件は脆弱性が実際に存在しており、2 件は誤検知であった。誤検知した 8 件の内 7 件は、ベンダーからの情報を参照した結果、RHEL7 系での影響が無いことが明記されているものであった。

また、8 件の内の 1 件は、脆弱性の存在する範囲が日付で指定されており、ソフトウェアのバージョン番号はセマンティックバージョンニング [5] に則ってつけられたものであったにも関わらず検知されていた。

最後に、LDAP サーバ 2 の修正履歴を基にソフトウェア情報修正機能が修正を行った後の SMTP サーバの結果を表 8 に示す。登録されたソフトウェア情報は 1366 件であり、新規のソフトウェア情報は 2 件であり、ソフトウェア情報修正機能を用いて修正されたものは 519 件あった。また、それ以外のソフトウェア情報は LDAP サーバで登録されたものと同じものであった。正常検知された 166 件、検知漏れの 1 件、誤検知の 8 件についても LDAP サーバの修正後と同じ結果となった。

表 6、表 7、表 8 の結果から再現率、適合率、F1 値を求めた結果を表 9 に示す。LDAP サーバの修正前の再現率は 0.556 であり、修正後は 0.994 であった。また、SMTP サーバの再現率は 0.994 であった。このことから、ソフトウェア情報修正機能を用いることで半分ほど検知漏れを起こしていた状況を解消することができ、脆弱性検知の再現率が向上することが確認できた。そのため、課題 1 を満たすことができたと言える。

4.2 利用者による評価

本節では、UI を実装することで利用者が運用可能であるかどうか評価を行う。

表 8 SMTP サーバの混同行列

	BEYOND で検知可能	BEYOND で検知不可能
脆弱性有り	166 件	1 件
脆弱性無し	8 件	1191 件

表 9 評価実験の結果

対象機器	再現率	適合率	F1 値
LDAP サーバの修正前	0.556	0.94	0.699
LDAP サーバの修正後	0.994	0.954	0.973
SMTP サーバ	0.994	0.954	0.973

4.2.1 評価方法

本節では、本学の情報システムの管理部局である情報メディアセンターの職員の方3名に実際にBEYONDを利用した後、アンケートによる評価を行った。質問項目を表10に示す。使いやすさといったユーザビリティの面について回答を求めた。質問項目に加えて、欲しい機能の自由記述の回答も求めた。

表10 ユーザビリティ評価のアンケート内容

項目番号	アンケート内容
1	脆弱性情報の検索、参照ページは使いやすいですか
2	機器情報の登録ページは使いやすいですか
3	機器情報の参照はしやすいですか
4	機器情報の修正機能は使いやすいですか

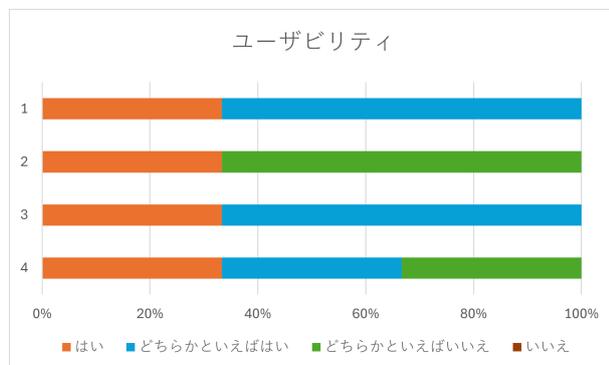


図7 ユーザビリティのアンケート結果

4.2.2 評価結果

結果を図7に示す。使いやすさについては、設問1, 3のように検索や参照を行う際の使いやすさについては、「はい」または「どちらかといえばはい」の回答が多かった。しかし、設問2, 4のように登録や修正を行う際の使いやすさについては、「どちらかといえばいいえ」の回答が多かった。自由記述回答には、WebインターフェイスだけでなくCSVファイルから一括修正などできたほうが良い、ExcelやCSVからのインポートの機能が欲しいといった回答が得られた。このことから、手動で情報を登録することの負担が大きいことがわかる。

また、情報をクローリングしている際にその状態が分からないので、表示待ちであるなら、そのような表示にしてほしいという意見があった。これは、現状のフロント部では、WebUIからWebAPIを操作するといった最低限の機能しか実装されていないため、UXに関する機能を追加していくことで解決出来る。

5 おわりに

本論では、ソフトウェア情報修正機能を追加することで、以前の課題であった再現率が不足していた問題を解決した。また、UIを開発することで本システムの各機構を統括し、利用者が運用できる状態となった。その一方で、システムを運用するにあたり、手動で情報を登録することに関しては負担が大きいことが分かった。そのため、今後の課題として、修正機能では、修正履歴から傾向を分析して同じような未対応パターンを一括で修正する機能が求められる。UIについても、情報の登録の負担を軽減するため、入力を選択式にする、入力項目を減らすといった負荷を軽減する機能が求められる。

参考文献

- [1] 独立行政法人情報処理推進機構 (IPA), “情報セキュリティ10大脅威2024解説書”, https://www.ipa.go.jp/security/10threats/nq6ept000000g22h-att/kaisetsu_2024.pdf, 2024/10/21.
- [2] IBM, “X-Force 脅威インテリジェンス・インデックス 2023”, <https://www.ibm.com/downloads/cas/QY8RQ2RK>, 2024/10/21.
- [3] 中村友昭, 竹原一駿, 大野真伯, 山下俊昭, 宗雪勝也, 小野滋己, 喜田弘司, 後藤田中, 最所圭三, “脆弱性情報を用いたセキュリティ保護システム“BEYOND”の開発”, 大学ICT推進協議会2022年度年次大会論文集, 13PM2B-3, 2022.
- [4] 中村友昭, 竹原一駿, 大野真伯, 山下俊昭, 宗雪勝也, 小野滋己, 喜田弘司, 後藤田中, 最所圭三, “BEYOND: セキュリティ対策システムの運用手法と脆弱性検知率向上のためのWebUIの開発”, 第86回全国大会講演論文集, Vol.1pp513-514, 2024.
- [5] Tom Preston-Werner, “セマンティックバージョンング 2.0.0 — Semantic Versioning”, <https://semver.org/lang/ja/>, 2024/10/21.