Computer Security Symposium 2020
26 - 29 October 2020

ウェブセキュリティガバナンスの実態調査

高田 雄太[1,a)]　熊谷 裕志[1]　神薗 雅紀[1]

概要：ウェブサイトは日に日に複雑性を増し，その管理運用は困難になっている．このような複雑なウェブサイトに対して定期的なメンテナンスを行い，セキュリティ対策を強化することは重要である．しかし，ウェブサイトの一部ページに残存する設定不備や脆弱性が未だに発見されており，それらを狙うサイバー攻撃は後を絶たない．そこで本稿では，セキュリティガバナンスの概念をウェブサイトに新たに適用し，その一環として，ウェブサイトに使用されている設定や技術の統一性を調査する．具体的には，同一ドメイン名の複数ページを分析し，ページ間における HTTP ヘッダのセキュリティ設定やウェブ技術のバージョンの差異を特定する．調査の結果，約 40%のウェブサイトに差異が特定され，中にはすべてのページで異なるバージョンの JavaScript ライブラリを使用するウェブサイトが存在した．このようなガバナンス不全に陥る原因を特定し，その改善策を検討する．

キーワード：セキュリティガバナンス，JavaScript ライブラリ，ウェブ観測

Measuring Web Security Governance

Yuta Takata[1,a)]　Hiroshi Kumagai[1]　Masaki Kamizono[1]

Abstract: While websites are becoming more and more complex daily, the difficulty of managing them is also increasing. It is important to conduct maintenance against these complex websites to strengthen their security. However, misconfigurations and vulnerabilities are still being discovered on some pages of websites and cyberattacks against them are never ending. In this paper, we newly apply the concept of security governance to websites, and, as part of this, measure the consistency of settings and technologies used on these websites. More precisely, we analyze multiple web pages with the same domain name and identify differences in the security settings of HTTP headers and versions of software among them. Our measurement results show that approximately 40% tested websites exhibit differences. For example, we found websites using a JavaScript library with different versions across all pages. We identify causes of such governance failures and propose the improvement plans.

Keywords: Security Governance, JavaScript Library, Web Measurement

1. はじめに

ウェブサイトは，企業組織を代表するシステムの一つであり，多くのユーザにサービスを提供している．したがって，サイバー攻撃によりウェブサイトが改ざんされたり，停止させられたりした場合の被害は大きい．このようなサイバー攻撃からウェブサイトを守るため，ウェブサイトの管理運用に関する様々なガイドラインが発行されている [1], [2]．これらガイドラインは，ウェブサイトのセキュリティ設定や脆弱性診断等の基本的なセキュリティ対策について記載しており，安心安全なウェブサイトの管理運用を支援する．しかしながら，ウェブサイトの設定不備や脆弱性は未だに発見されており，ウェブサイトに対するサイバー攻撃は後を絶えない [3]．

企業組織の規模拡大に伴い，ウェブサイトが提供するサービスの数も増加する．このようなウェブサイトでは，サービスを迅速に開発するため，しばしばサードパーティ

[1] デロイト トーマツ サイバー合同会社
Deloitte Tohmatsu Cyber LLC
[a)] yuta.takata@tohmatsu.co.jp

© 2020 Information Processing Society of Japan

コンテンツが活用されている [4]．しかし，その結果としてサードパーティへの依存が大きくなり，ウェブサイトの複雑度が増加する．この複雑度が，ウェブサイトの一部ページにおける対策漏れや新たな攻撃経路を顕在化させる [5], [6]．例えば，Lauinger らは，同一ページに異なるバージョンの jQuery を複数参照したウェブサイトを発見している．彼らはこのようなライブラリの重複参照は，サードパーティコンテンツの利用が発生原因の一つであり，脆弱性の発現に影響を与えると報告している [7]．これら個別のセキュリティ課題への対応はもちろん重要だが，それらセキュリティ対策はウェブサイトの単一ページではなく，すべてのページに適用すべきである．すなわち，このような複雑なウェブサイトに対して，ガバナンスを効かせることにより，すべてのページに渡って一貫したセキュリティ対策を講じることが重要であると言える．

本稿では，セキュリティガバナンスの概念をウェブサイトに新たに適用し，その一環として，ウェブサイトに使用されている設定や技術の統一性を調査する．主なモチベーションは，単一ではなくすべてのページにメンテナンスやセキュリティ対策が浸透しているかを調査することにある．この**ウェブセキュリティガバナンス**の実態を明らかにすべく，日本国内企業組織のウェブサイトを対象に，同一ドメイン名を持つ複数の異なるページにアクセスし，それらページ間で共通して使用された設定や技術の差異を分析する．本稿の貢献は以下のとおりである．

- セキュリティガバナンスの概念を活用し，ウェブサイトにおけるセキュリティ対策の統一性を調査する．
- 国内企業組織のウェブサイト 2,817 件を調査し，ページ間で使用されている設定や技術に差異のあるウェブサイトが 1,119 件 (39.7%) 存在することを示す．
- jQuery のバージョン差異が 955 件 (85.3%) と最も多く，すべてのページで異なるバージョンを使用するウェブサイトが存在することを示す．
- ウェブサイトに使用されている設定や技術が不統制となる原因を特定し，各原因に対する改善策を示す．

2. 研究背景

2.1 ウェブセキュリティ対策

安全にウェブサイトを運用管理するためのセキュリティ対策について，様々なガイドラインが発行されている [1], [2]．ガイドラインでは，OS，サーバソフトウェア，ミドルウェア等のウェブサーバや，JavaScript ライブラリ，フレームワーク，CMS (Content Management Systems) 等のウェブアプリケーションに対する定期的なセキュリティチェックを推奨している．これらサーバやアプリケーションにおける既知脆弱性の悪用を防ぐために，ウェブサイトを構成する技術を確認し，必要に応じてバージョン更新やパッチ適用等の継続的なメンテナンスが重要である．

上記のセキュリティ対策に加え，ウェブサイトの HTTP 応答ヘッダにより，ブラウザのセキュリティ機能を活用する対策もある [8]．例えば，`Strict-Transport-Security` や `Content-Security-Policy` 等がある．これらヘッダ (以降では，総称して “HTTP セキュリティヘッダ” と呼ぶ) を設定することにより，ブラウザに HTTPS 通信の利用を促進したり，コンテンツインジェクション攻撃を防止したりできる．サーバ側の対策だけでなく，このようなクライアント側の対策も併用することにより，より一層効果的なセキュティ対策を実現できる．

2.2 ウェブセキュリティガバナンス

前節のウェブセキュリティ対策の効果を最大化するため，組織内でウェブサイトの運用管理に関する基準や方法，手順を統一することは重要である．企業組織における情報資産に係るリスク管理や情報セキュリティに関わる意識醸成や説明責任を徹底させるセキュリティガバナンスのように，ウェブサイトにもガバナンスを効かせることにより，セキュリティ対策漏れを軽減できる．例えば，ウェブ技術はそのバージョンによって，脆弱性の有無や特徴が変化する．このような脆弱性の悪用リスクを軽減するため，ウェブサイトのすべてのページにおいて，どの技術のどのバージョンを使用するかを統一管理することが望ましい．ガバナンスを効かせてサーバ設定やウェブ技術を統一することにより，対策漏れの軽減に加え，新しい対策技術の導入や新しい脆弱性情報の対応の促進，すなわちレジリエンスの向上も期待できる．しかしながら，ウェブサイトの設定不備や脆弱性を悪用したサイバー攻撃は依然として発生している．これは企業組織がガバナンス不全に陥ったことにより，ウェブセキュリティ対策にも影響が及んだことが原因の一つであると考えられる．

そこで我々は，このウェブセキュリティガバナンスの一環である，サーバ設定やウェブ技術の複数ページにおける統一性を調査した．企業組織のウェブサイトがガバナンス不全に陥る原因を特定するとともに，その改善策を検討する．ウェブサイトの各ページにおけるセキュリティ対策を分析した既存研究は数多く存在するが，複数ページの分析により初めて分かるセキュリティ対策の統一性 (ガバナンス) を調査した研究は，我々の知る限り存在しない．

3. 既存研究

3.1 ウェブサイトの観測調査

これまで多くの研究者により，ウェブサイトにおけるセキュリティ対策や技術の普及に関する調査がなされてきた．Van Goethem らは，欧州のウェブサイトにおける脆弱性やウェブセキュリティ対策の状況を大規模に調査し，国やウェブサイト人気度と対策状況との間の関係を分析している [9]．その他，HTTP セキュリティヘッダや JavaScript

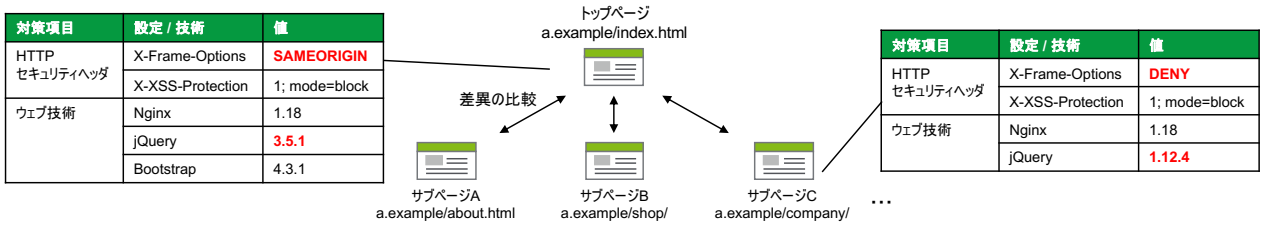

図 1　ウェブセキュリティガバナンス分析

ライブラリ等，クライアント側の技術の変遷についてインターネットアーカイブを用いて調査した研究 [5] やサードパーティコンテンツ依存関係に起因する HTTPS 通信への悪影響を評価した研究 [6] がある．いずれの研究もウェブセキュリティ対策の普及やその設定不備といった個別の事象を分析しているが，本研究はそれら各種対策の統一性に着目した分析を目的としている．

3.2　HTTP セキュリティヘッダの分析

HTTP セキュリティヘッダは，設定するだけで一定のセキュリティ対策効果が得られる．しかしながら，ブラウザ間でこれらヘッダに対する解釈に矛盾が生じていると報告されている．例えば，`Content-Security-Policy` ヘッダや `X-FrameOptions` ヘッダに着目した研究があり，確認された矛盾の詳細やセキュリティへの影響が報告されている [10], [11]．本研究の分析においても，これら矛盾に関連する結果を確認しており，上記既存研究の成果を補完する (5.2.2 節を参照)．

3.3　ウェブ技術の分析

ウェブサーバやアプリケーション等の様々なウェブ技術の脆弱性や不備を分析した多くの研究が存在する．Vasek らは，使用されているウェブ技術の種類やバージョンがウェブサイト改ざんのリスク要素となり得るかケースコントロール研究を実施している [12]．その他，サードパーティ JavaScript ライブラリや既知脆弱性のあるライブラリの使用と，そのセキュリティへの影響についても調査されている [4], [7]．しかしながら，上記の研究はいずれも，ウェブサイトにおける単一ページを分析しており，複数のページを横断的に分析していない．

4.　ウェブセキュリティガバナンスの実態調査

ウェブサイトの複数ページ間におけるセキュリティ対策の差異を比較することにより，ウェブセキュリティガバナンスの実態を調査する．具体的には図 1 に示すように，同一ドメイン名における異なるページの間で共通して使用された設定および技術における値を比較し，その差異を特定する．

4.1　URL の収集

ウェブサイトのルートディレクトリに当たる URL や検索エンジン等に “公式サイト” として掲載されている URL は，管理者により注意深くメンテナンスされているページ (トップページと呼ぶ) だと考えられる．そこで本分析では，そのようなトップページを基準に，トップページと同一のドメイン名を持つ異なる URL のページ (サブページと呼ぶ) を複数選定する．サブページは，検索エンジンによる検索結果やトップページの内部リンクとし，トップページと同一のドメイン名を持つ異なるパスの URL となる．なお，本分析では，PDF ファイル等のメディアコンテンツを除外し，HTML コンテンツのみを対象とする．

4.2　ガバナンスの分析

トップおよびサブページの URL にアクセスし，収集したデータに基づき，ウェブセキュリティ対策の差異を特定する．対策の指標として，2.1 節に記載した HTTP セキュリティヘッダの設定およびウェブ技術のバージョンを用いる．図 1 の例では，本分析により，トップページおよびサブページ C の間で，HTTP セキュリティヘッダ `X-Frame-Options` の設定およびウェブ技術 jQuery のバージョンが差異として特定される．

4.2.1　HTTP セキュリティヘッダの分析

HTTP セキュリティヘッダは，その性質からセキュリティ対策の現状を把握するための指標としても用いられている．特に，`Strict-Transport-Security`, `Content-Security-Policy`, `X-Frame-Options`, `X-XSS-Protection`, `X-Content-Type-Options` の 5 つの HTTP 応答ヘッダは，研究者によってその普及率や設定内容が調査されている [5], [9], [11], [13]．本分析においても，これらのヘッダをウェブセキュリティ対策の指標として検知し，トップおよびサブページ間で共通して使用されたヘッダの設定値を比較する．

4.2.2　ウェブ技術の分析

ウェブサイトを構成するサーバやアプリケーション等のウェブ技術とそのバージョンは，静的解析と動的解析の二種類のアプローチで検知できる [7]．

静的解析． ウェブサイトアクセス時に観測した URL や HTTP ヘッダ，HTML コンテンツに対して，正規表現や

表 1　ウェブサイトアクセス結果

アクセス結果	トップページ	サブページ
HTTP 200 OK	2,817	13,270
HTTP エラー	3	45
アクセスエラー	36	127
合計	2,856	13,442

キーワードでマッチングすることにより，ウェブ技術を検知する方法である．バージョン値は，一定のパターンで記載されていた場合に抽出できる．

動的解析．ウェブサイトに含まれる JavaScript を実行し，グローバルスコープに定義されたデータを分析することにより，ウェブ技術を検知する方法である．バージョン値は，グローバルスコープにおける特定の変数またはオブジェクトに格納されていた場合に抽出できる．

本分析では，上述した二つの方法でウェブ技術を検知し，トップおよびサブページ間で共通して使用された技術のバージョン値を比較する．

4.3　調査環境

企業組織のウェブセキュリティガバナンスを調査するために，調査対象から個人のウェブサイトを除外する必要がある．そこで本調査では，日経会社情報 DIGITAL [14] の業界/企業一覧に掲載されている日本国内企業の URL 合計 2,856 件をトップページとして使用した．また，サブページの URL として，検索エンジン Bing およびクエリパラメータ “site” を用いて，トップページのドメイン名を検索した結果のリンクを使用した．なお，検索結果が不十分であった場合は，トップページの内部リンクも使用し，最大で 5 件のサブページ URL を収集した．

収集した URL にアクセスし，各ページの HTTP ヘッダやコンテンツを収集するために，Ubuntu にインストールした Chromium [15] を用いた． コンテンツを読み込んだ後，JavaScript の実行や非同期通信が発生することを考慮し，ネットワーク接続がアイドル状態になるまで待機した．なお，3 分間でコンテンツの読み込みが終わらない場合は，アクセスをタイムアウトした．HTTP セキュリティヘッダの検知には，入力した URL または転送先の URL の応答 HTTP ヘッダを用いた．ウェブ技術の検知には，4.2.2 節に記載したアプローチで多彩な技術を検知できるオープンソースの Wappalyzer [16] を用いた．

5.　調査結果

5.1　URL の収集結果

2020 年 4 月にトップページの URL にアクセスした結果，表 1 に示すとおり，2,817 件が正常に応答した．それ以外は，HTTP ステータスコード 400 番台や 500 番台の HTTP エラー，DNS 名前解決エラー，またはタイムアウト等のアクセスエラーを応答した．加えて，トップページのドメイン名に基づき収集したサブページ URL は 13,442 件あり，その内 13,270 件が正常にコンテンツを応答した．この時，検索結果が 5 件に満たないウェブサイトは 113 件あり，その内 80 件のウェブサイトにおいて，内部リンクからサブページ URL を補填できた．

表 2　トップ/サブページ間で特定された差異

差異	件数	割合 (N=2,817)
HTTP セキュリティヘッダの設定値	6	0.2%
ウェブ技術のバージョン値	1,091	38.7%
ヘッダおよび技術の両方	22	0.8%
差異なし	1,572	55.8%
サブページなし	126	4.5%

表 3　トップページで検知された HTTP セキュリティヘッダ

ヘッダ名	件数	割合 (N=2,817)
X-Frame-Options	528	18.7%
X-Content-Type-Options	347	12.3%
X-XSS-Protection	337	12.0%
Strict-Transport-Security	229	8.1%
Content-Security-Policy	32	1.1%

表 4　サブページで異なる値が設定された HTTP セキュリティヘッダ

ヘッダ名	二重設定	異なる値	設定ミス
X-Frame-Options	12	3	0
X-Content-Type-Options	9	0	0
X-XSS-Protection	4	1	0
Strict-Transport-Security	1	2	1

5.2　ガバナンスの分析結果

5.2.1　複数のページ間における差異

複数のページ間における HTTP セキュリティヘッダの設定値およびウェブ技術のバージョン値の差異を分析した．その結果，表 2 に示すとおり，1,119 件 (39.7%) のウェブサイトにおいて何らかの差異が特定された．これらの差異は，ほとんどがウェブ技術のバージョン値の差異 (詳細は 5.2.3 節に記載) であり，HTTP セキュリティヘッダの設定値の差異はわずか 28 件 (詳細は 5.2.2 節に記載) であった．それ以外は，ページ間で “差異なし” のウェブサイトが 1,572 件 (55.8%)，トップページのみで “サブページなし” のウェブサイトが 126 件 (4.5%) であった．

“サブページなし” のウェブサイトは，検索結果や内部リンクのないウェブサイトに加えて，トップページのドメイン名と異なるサブドメイン名を使用したリンクや PDF ファイルへのリンクで構成されたウェブサイトであった．本分析は，トップページと同一ドメイン名を持つサブページおよび HTML コンテンツを分析対象としたため，これらウェブサイトは “サブページなし” に分類された．

5.2.2　HTTP セキュリティヘッダの設定値の差異

トップページで検知された HTTP セキュリティヘッダを

表 5　トップページで検知されたウェブ技術 Top10

技術名	件数	割合 (N=2,817)
jQuery	2,666	94.6%
Apache	500	17.8%
IIS	398	14.1%
Microsoft ASP.NET	327	11.6%
jQuery UI	282	10.0%
Nginx	278	9.9%
Modernizr	274	9.7%
Underscore.js	180	6.4%
WordPress	171	6.0%
jQuery Migrate	170	6.0%

表 6　サブページで異なるバージョンが使用されたウェブ技術 Top10

技術名	カテゴリ	件数
jQuery	JavaScript ライブラリ	955
Nginx	ウェブサーバ	152
Apache	ウェブサーバ	120
WordPress	CMS	42
Apache Traffic Server	ウェブサーバ	32
IIS	ウェブサーバ	32
Jetty	ウェブサーバ	26
lighttpd	ウェブサーバ	25
Microsoft ASP.NET	ウェブフレームワーク	19
jQuery UI	JavaScript ライブラリ	10

表 3 に示す．`X-Frame-Options` ヘッダが，528 件 (18.7%) と最も多く設定されていた．このヘッダは，フレームを通じて他ウェブサイトのコンテンツが含まれないよう保証することにより，クリックジャッキング攻撃を防ぐ目的で使用される．しかしながら，`X-Frame-Options` 単体では二重のフレームを用いたクリックジャッキング攻撃に対して有効ではないため，Calzavara らは `Content-Security-Policy` ヘッダと組み合わせて使用することを推奨している [11]．`Content-Security-Policy` ヘッダは，読み込むリソースを制限することにより，クロスサイトスクリプティング攻撃やコンテンツインジェクション攻撃を防ぐ．しかしながらこのヘッダは，その管理運用の煩雑さから設定ミスの多いヘッダであり，設定しているウェブサイトは限定的であると報告されている [10], [13]． 本分析においても，わずか 32 件 (1.1%) のウェブサイトでしか設定されていなかった．

上述したトップページの HTTP セキュリティヘッダと比較して，異なる値が設定されたサブページのヘッダを表 4 に示す．これら差異の内訳は，同じ値が多重設定されたヘッダが 26 件，異なる値が設定されたヘッダが 6 件，セミコロン抜けのヘッダが 1 件であった．同じ値が多重設定されたヘッダは，例えば，“`X-Frame-Options: SAMEORIGIN SAMEORIGIN`” のようなヘッダだが，ブラウザによって動作が異なり矛盾をきたすと報告されている [11]．

このような HTTP セキュリティヘッダの差異の原因は，サーバやアプリケーションにおける設定が統一されていないことやネットワークアプライアンス等の中間装置によるヘッダ挿入等が考えられる [17]．

5.2.3　ウェブ技術のバージョン値の差異

トップページでバージョン検知されたウェブ技術の集計結果を表 5 に示す．検知数トップの jQuery は，2020 年 8 月現在，グローバルのウェブサイトにおける使用率 76.2% を誇る人気の JavaScript ライブラリである [18]．日本国内企業のウェブサイトでは，グローバルにおける割合よりも約 20%高く 94.6%のウェブサイトで使用されており，より影響力の強いウェブ技術であることが分かる．

上述したトップページのウェブ技術と比較して，異なるバージョン値が使用されたサブページのウェブ技術を表 6 に示す．jQuery が 955 件と最も多く，35.8% (N=2,666) のウェブサイトにおいてバージョンが統一されていないことを意味する．一方，続く Nginx や Apache といったウェブサーバの差異は，サードパーティコンテンツが主な原因であった．各ページで使用された広告やアクセス解析等のサードパーティコンテンツが異なったため，そのコンテンツがホストされているウェブサーバのバージョン値も同様に異なった．また，CMS である WordPress は，バージョン値の削除による差異がほとんどであったが，ページ間で異なるバージョンの WordPress を使用していたウェブサイトも 5 件だけ存在した．ページ単位でデザインやコンテキストを切り替える等の理由により，このようなアプリケーションレイヤーの差異が生じると考えられる．

CMS はバージョンの違いにより管理パネル UI も変化することが多く，比較的ウェブサイト管理者は気づきやすい．一方 jQuery は，バージョンの違いに起因する視覚的な変化がほとんどないため，管理者の意図していないバージョン差異が生じている可能性がある．そこで以降では jQuery に着目し，バージョンの普及や差異，検知方法，参照方法の詳細を分析する．

5.3　jQuery ガバナンス

5.3.1　jQuery のバージョン

トップおよびサブページにおいて検知した jQuery バージョンの集計結果を図 2 に示す．検知したメジャーバージョンは，3.x 系よりも 1.x 系が多いことが分かる．最も多く検知した jQuery 1.12.4 は，2020 年 8 月現在 1.x 系の最新バージョンであり，WordPress が後方互換性を持たせるために使用されていることが知られている [19]．WordPress のような CMS では，サードパーティ開発者により作成された多くのプラグインが jQuery を使用している．後方互換性を持たせないとこれらプラグインが動作しなくなり，多くのウェブサイトが影響を受ける．そのため，このような古いバージョンが使用されていると考えられる．

その他，同一ページ内で複数の異なるバージョンの jQuery

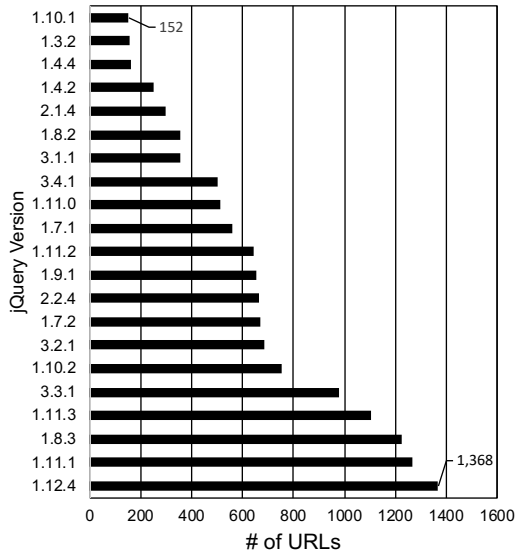

図 2 検知した jQuery のバージョンとその件数 ($N >= 150$ を対象)

表 7 jQuery バージョンの種類数

種類数	ウェブサイト件数	割合 (N=955)
2 種類	665	69.6%
3 種類	203	21.3%
4 種類	67	7.0%
5 種類	16	1.7%
6 種類 (バラバラ)	4	0.4%

を参照していたウェブサイトが 358 件存在した．このような重複参照は，参照そのものの自動/手動設定の混在や，サードパーティコンテンツの混在等が原因と考えられているが，ウェブサイトにエラーが発生しない限り，管理者は気づかないのが現状である．しかしながら，重複参照はウェブサイトの意図しない動作を招いたり，脆弱性の発現に影響を与えたりするため，改善すべきである [7]．

5.3.2 jQuery のバージョン差異

5.2.3 節で特定した 955 件の jQuery バージョン差異は，ある 1 ページだけが異なるのか，もしくは複数ページでバラバラに異なるのかを特定するために，jQuery バージョンの多様性を分析した．トップページ 1 件およびサブページ 5 件の最大 6 ページに使用された jQuery のバージョン種類数を集計した結果を表 7 に示す．驚くべきことに 6 ページすべてにおいて異なるバージョンを使用していたウェブサイトが 4 件存在した．これらウェブサイトには，各ページの内容やデザインはバラバラで，URL パスの第一階層が異なるという特徴があった．ページの内訳は，リクルートサイトやティザーサイト，支店や店舗，商材の紹介ページ，ニュースや解説記事等であった．前述した WordPress の差異と同様にウェブサイトのコンテキストを切り替える場合や，ページごとにウェブサイト管理者が異なる場合に，このような差異が生じると推測される．

表 8 動的解析のみでバージョン検知できた jQuery の使用

使用ページ数	ウェブサイト件数	割合 (N=955)
0 ページ	311	32.6%
1 ページ	425	44.5%
2 ページ	181	19.0%
3 ページ	31	3.2%
4 ページ	7	0.7%
5 ページ	0	0.0%
6 ページ	0	0.0%

表 9 jQuery の参照元 (Referer) と参照先 (Host)

参照元 (Referer)	参照先 (Host)	件数
ファーストパーティ	ファーストパーティ	3,659
ファーストパーティ	サードパーティ	1,765
サードパーティ	ファーストパーティ	0
サードパーティ	サードパーティ	68

5.3.3 jQuery の検知方法

Wappalyzer は，URL の静的解析および JavaScript の動的解析により jQuery とそのバージョンを検知する．静的解析は，明示的な情報を含む URL であれば jQuery のバージョンを検知できるが，`https://a.example/lib/jquery.js` のような URL だとバージョンを検知できない．一方の動的解析は，URL に明示的な情報がなくても，最後に読み込まれた jQuery のバージョンを検知できる．このような動的解析のみで検知できた jQuery は，644 件 (67.4%, N=955) のウェブサイトで使用されており，表 8 に示すとおり，複数ページで使用されていた．これら暗黙的に使用された jQuery がバージョン管理に影響を与えるか特定するため，表 8 の合計 4 ページで使用されたウェブサイト 7 件を手動で分析した．その結果，これらウェブサイトは，バージョン値のない URL に加え，複数の JavaScript ライブラリがバンドルされたファイルを用いて jQuery を使用していた．ライブラリのバンドルファイルは，webpack [20] 等のモジュールバンドラにより管理できるが，バンドルファイルに加え，他にも異なるバージョンの jQuery を使用したページが存在した．したがって，開発者がバンドルされた jQuery または手動で導入した jQuery の存在を忘れたことが原因で，両方を使用してしまったと推測できる．

5.3.4 jQuery の参照方法

jQuery の参照方法は，公式サイトや npm のようなパッケージマネージャからダウンロードし，自身のサーバ (ファーストパーティ) に保存した jQuery を使用する方法 [21], [22] と，Google や Microsoft 等が運用する外部サーバ (サードパーティ) に保存された jQuery を使用する方法 [23], [24] の二種類ある．これら jQuery の参照方法の違いがバージョン差異につながるか特定するため，jQuery の参照元 (Referer) および参照先 (Host) を集計した．個々の参照の集計結果を表 9 に示す．ファーストパーティにホストされた jQuery の参照はサードパーティのそれよりも

表 10　本分析で特定したガバナンス不全に対する改善策

改善策	特定したガバナンス不全
ガイドラインの規定	アプリケーションレイヤーの差異 (5.2.3)
	コンテキストや管理者の違い (5.3.2)
セキュリティ対策の自動化	HTTP ヘッダの多重設定 (5.2.2)
	独自バンドルファイルの使用 (5.3.3)
	ライブラリ参照先の混在 (5.3.4)
サプライチェーンマネジメント	サードパーティの把握困難 (5.2.3, 5.3.4)
	サードパーティによる技術制約 (5.3.1)
定期的な第三者評価	上記すべて

二倍多かった．この内，jQuery の参照先 (ファーストパーティまたはサードパーティ) をページ間で統一していないウェブサイトが，407 件 (42.6%, N=955) 存在した．この参照先の混在は，jQuery バージョン差異のなかったウェブサイトでは，106 件 (5.7%, N=1,862) しか確認されず，jQuery の参照方法の違いが，jQuery のバージョン差異の一因となっていることを示唆している．

また，サードパーティ参照元の jQuery が 68 件あることが分かる．これは広告や SNS メディア等のサードパーティコンテンツを読み込むウィンドウやフレームが使用され，その内部で jQuery が参照されたことを意味する．このようなサードパーティコンテンツは，ウェブサイト管理者による把握が困難であり，意図しない参照となる可能性があるため，注意が必要である [6]．

6. 議論

6.1 ガバナンス不全の改善

前章で特定したガバナンス不全に対して，表 10 にす 4 つの改善策を提案する．

ガイドラインの規定．ウェブサーバの調達や設定，ウェブサイトの開発や運用の基準を統一するためには，組織内におけるガイドラインの規定および遵守が有効である．業務内容や取り扱う情報，保有する情報システムに関するリスク評価の結果を踏まえた上でガイドラインを整備することにより，現状に沿ったセキュリティ対策を適用できる．特に本策は，コンテキストや管理者が異なるウェブサイトを持つ組織に適用することにより，ガバナンス不全の改善を期待できる．ウェブ技術の発展やウェブサイトの役割は変化が激しいため，ガイドラインの陳腐化や形骸化を防ぐための定期的な見直しも重要である [25]．

セキュリティ対策の自動化．手動による管理運用が原因のガバナンス不全に対する直接的な改善策は，その自動化である．本策は，SecDevOps とも呼ばれ，DevOps にセキュリティの要素を取り入れることにより，機能追加やシステムリリース時に一定のセキュリティ対策を自動化できる．DevOps は，開発と運用，品質維持を協調して行い，システムの価値向上ならびにその価値を迅速にユーザへ提供するための思想文化である．DevOps のためのツールを導入することにより，ウェブサイトの環境構築，ビルド，デプロイといった一連のプロセスを自動化できる [26]．SecDevOps は，この DevOps に，ロギングや監視，セキュアコーディング，テスト等のセキュリティ対策を統合し，機能追加やウェブサイト公開時のセキュリティチェックを自動化できる [27]．本分析で特定したガバナンス不全の中では，HTTP ヘッダの多重設定や独自バンドルファイルの使用，ライブラリ参照先の混在の改善が期待できる．

サプライチェーンマネジメント．管理者がサードパーティコンテンツの内容や依存関係を把握することは，予期せぬ設定不備や脆弱性の発現を防ぐために重要である [4], [6]．オンデマンドで参照するサードパーティコンテンツに加えて，あらかじめダウンロードした JavaScript ライブラリ等のサードパーティコンテンツの管理も重要である．これらの管理には，前述のように，自動化のためのツールを導入すべきである [7]．例えば，JavaScript のパッケージマネージャ npm は，重複参照を予防できるとともに，どのライブラリのどのバージョンをどこからダウンロードして使用しているか一括管理できる [22]．

ウェブサイトの開発運用をサードパーティ事業者に委託した場合であっても，前述したガイドラインや使用すべき技術やツール，サードパーティコンテンツ等の要求事項を仕様書等に定めるべきである．サードパーティ事業者においても同レベルのセキュリティ対策が確実に実施されるようガバナンスを効かせることが重要である [28]．

定期的な第三者評価．ガバナンス不全を未然に防ぐために，SecDevOps によるセキュリティチェックに加え，独立性を有する第三者によるセキュリティチェックも重要である．本調査や類似のチェックを定期的に実施することにより，基準に準拠しているか，セキュリティ対策は有効に機能しているか，各対策の状況を評価する [29]．

6.2 互換性の問題

本調査では，HTTP セキュリティヘッダの設定やウェブ技術のバージョンの統一性を分析した．しかしながら，中には互換性を保つことを理由に設定やバージョンを統一することが困難なウェブサイトも存在する [2]．特に jQuery は，マイナーバージョン間でさえも後方互換性が欠如するケースがあり，バージョン更新を阻害する要因として問題視されている [7]．このような互換性の問題によりガバナンス不全となったウェブサイトは，改修に長時間を要する場合があるが，その役割や重要性を踏まえ今後の対応を決定すべきである． もし不要なウェブサイトである場合は，統廃合する対応も考えられる [30]．

6.3 制限事項

本分析では，同一ドメイン名を持つページは同一組織が管理がしていると考え，それらページ間に限った差異を特

定している．しかしながらウェブサイトの中には，5.2.1 節に記載したようなサブドメイン名を多く使用したウェブサイトも存在する．どの範囲までウェブセキュリティガバナンスを効かせるかは各企業組織によって戦略が異なるが，サブドメイン名も含めた範囲の分析も有益であると考えられるため，今後の課題としたい．

ウェブ技術の検知における静的解析および動的解析は，5.3.3 節に記載のとおり，失敗するケースがある．例えば，Wappalyzer [16] が対応していないウェブ技術や開発者が独自に改良を加えたウェブ技術は検知できない．しかしながら，JavaScript ライブラリトップシェアの jQuery やその他シェア上位の技術を検知できていたことから，本分析による見逃しが本調査結果に与える影響は小さいと考える．

本研究では，日本国内企業組織に関連するウェブサイトを対象に，ウェブセキュリティガバナンスを調査した．国内における局所的な結果であるため，グローバル企業組織のウェブサイトを対象とした調査は今後の課題としたい．

6.4 研究倫理

本調査では，ウェブサイトの複数ページにアクセスし，データを収集している．この時，特定のウェブサイトに負荷が集中しないよう，アクセスや時間帯を分散させるとともに，アクセス後も短い滞在時間となるよう配慮した．また，本調査により特定の企業組織にネガティブな影響を与えないよう，具体的なドメイン名や企業組織名は記載せず，多くのウェブサイトを分析した．

7. おわりに

複雑化するウェブサイトに対して，セキュリティガバナンスを新たに適用し，その一環として，ウェブサイトに使用されている設定や技術の統一性を調査した．ガバナンスを効かせウェブサイトを制御することにより，セキュリティ対策漏れを予防でき，結果としてレジリエンスの向上に繋がる．本稿の調査結果が，定期的なウェブサイトの棚卸しや見直し，そしてセキュリティガバナンス強化のきっかけになることを期待する．

参考文献

[1] NIST, “SP 800-44: Guidelines on Securing Public Web Servers.” https://doi.org/10.6028/NIST.SP.800-44ver2.

[2] IPA, “安全なウェブサイトの運用管理に向けての 20 ヶ条〜セキュリティ対策のチェックポイント〜.” https://www.ipa.go.jp/security/vuln/websitecheck.html.

[3] IPA, “10 Major Security Threats 2019.” https://www.ipa.go.jp/files/000076989.pdf.

[4] N. Nikiforakis *et al.*, “You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions,” in *ACM CCS*, 2012.

[5] B. Stock *et al.*, “How the Web Tangled Itself: Uncovering the History of Client-Side Web (In) Security,” in *USENIX Security Symposium*, 2017.

[6] D. Kumar *et al.*, “Security Challenges in an Increasingly Tangled Web,” in *WWW*, pp. 1–8, 2017.

[7] T. Lauinger *et al.*, “Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web,” in *NDSS*, 2017.

[8] “OWASP Secure Headers Project.” https://owasp.org/www-project-secure-headers/.

[9] T. Van Goethem *et al.*, “Large-scale security analysis of the web: Challenges and findings,” in *TRUST*, pp. 110–126, 2014.

[10] S. Roth *et al.*, “Complex Security Policy? A Longitudinal Analysis of Deployed Content Security Policies,” in *NDSS*, 2020.

[11] S. Calzavara *et al.*, “A Tale of Two Headers: A Formal Analysis of Inconsistent Click-Jacking Protection on the Web,” in *USENIX Security Symposium*, 2020.

[12] M. Vasek *et al.*, “Hacking is not random : a case-control study of webserver compromise risk,” *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 206—-219, 2015.

[13] S. Calzavara and M. Bugliesi, “Content Security Problems? Evaluating the Effectiveness of Content Security Policy in the Wild,” in *ACM CCS*, pp. 1365–1375, 2016.

[14] “日経会社情報.” https://www.nikkei.com/nkd/.

[15] “The Chromium Projects.” https://www.chromium.org/Home.

[16] “Wappalyzer.” https://www.wappalyzer.com/.

[17] G. Tyson *et al.*, “Exploring HTTP Header Manipulation In-The-Wild,” in *WWW*, 2017.

[18] “Usage statistics of JavaScript libraries for websites.” https://w3techs.com/technologies/overview/javascript_library.

[19] “Why wordpress only use old jQuery version is 1.12.4?.” https://wordpress.org/support/topic/why-wordpress-only-use-old-jquery-version-is-1-12-4/.

[20] “webpack.” https://webpack.js.org/.

[21] “jQuery CDN.” https://code.jquery.com/.

[22] “npm.” https://www.npmjs.com/.

[23] “Hosted Libraries — Google Developers.” https://developers.google.com/speed/libraries.

[24] “Microsoft Ajax Content Delivery Network.” https://docs.microsoft.com/en-us/aspnet/ajax/cdn/overview.

[25] NIST, “SP 800-35: Guide to Information Technology Security Services.” https://doi.org/10.6028/NIST.SP.800-35.

[26] C. Ebert *et al.*, “DevOps,” *IEEE Software*, vol. 33, pp. 94–100, 2016.

[27] V. Mohan and L. Ben Othmane, “SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps,” in *ARES*, pp. 542–547, 2016.

[28] NISC, “外部委託等における情報セキュリティ上のサプライチェーン・リスク対応のための仕様書策定手引書.” https://www.nisc.go.jp/active/general/pdf/risktaiou28.pdf.

[29] NISC, “情報セキュリティ監査実施手順の策定手引書.” https://www.nisc.go.jp/active/general/pdf/SecurityAuditManual.pdf.

[30] IPA, “サーバソフトウェアが最新版に更新されにくい現状および対策.” https://www.ipa.go.jp/security/technicalwatch/20140425.html.