HTML5 アプリケーションのセキュリティリスクに関する一考察

奥田 哲矢† 知加良 盛† 栢口 茂†

†NTT セキュアプラットフォーム研究所
180-8585 東京都武蔵野市緑町 3-9-11
okuda.tetsuya@lab.ntt.co.jp, chikara.sakae@lab.ntt.co.jp, kayaguchi.shigeru@lab.ntt.co.jp

あらまし 近年，主要 WEB ブラウザが対応を進める HTML5 は，WEB アプリケーションに新たな機能・操作性を提供可能である一方，未知のセキュリティリスクを抱えている可能性がある．今後，HTML5 関連 API を利用した WEB アプリが普及することで，HTML5 特有の攻撃対象の増加，攻撃経路の拡大が想定される．このため，アプリ開発においては，利用する API に対するリスクの把握が必要である．本研究では，既存の HTML5 セキュリティレポートを分析し，アプリ開発者が対策すべきリスク（計 76 項目）を抽出した．本論文では，リスク抽出の過程で得られた，各 API が抱えるリスクの傾向，リスクの分類体系，及びセキュリティモデルについて記述する．

Security Risks of HTML5 Web Application

Tetsuya Okuda† Sakae Chikara† Shigeru Kayaguchi†

†NTT Secure Platform Laboratories
3-9-11 Midori-cho, Musashino-shi, Tokyo
okuda.tetsuya@lab.ntt.co.jp, chikara.sakae@lab.ntt.co.jp, kayaguchi.shigeru@lab.ntt.co.jp

Abstract HTML5 web application is superior in functionality and operability, however, the security risk of HTML5 web application is unclear. As HTML5 web applications become more common, the increase of attack is expected. That is why web application developers should be aware of the risks of HTML5 web applications. We surveyed about the risks to be understood by web application developers, by analyzing the existing security report. In this paper, we present the tendency of risks of each APIs, a classification framework of risks, and a security model of HTML5 web application.

1 はじめに

近年，主要 WEB ブラウザが対応を進める HTML5 は，WEB アプリケーションに新たな機能・操作性を提供可能である一方，未知のセキュリティリスクを抱えている可能性がある．今後，HTML5 関連 API を利用した WEB アプリが普及することで，HTML5 特有の攻撃対象の増加，攻撃経路の拡大が想定される．このため，アプリ開発においては，利用する API に対するリスクの把握が必要である．

本研究では，既存の HTML5 セキュリティレポート(ENISA[1],OWASP[2])を分析し，アプリ開発者が対策すべきリスク（計 76 項目）を抽出した．このうち本論文では，リスク抽出の過程で得られた，各 API が抱えるリスクの傾向，リスク

の分類体系，及びセキュリティモデルについて記述する．以降，HTML5 関連 API を利用する WEB アプリケーションを HTML5 アプリケーションと呼ぶ．

2　セキュリティモデル

本研究では，アプリ開発者に「何を何故守るのか」というセキュリティ観点を伝えるため，一般の WEB アプリのセキュリティモデル[3]を参考にして，HTML5 特有の攻撃対象及び攻撃経路を考慮したセキュリティモデルを構築した（図1）．セキュリティモデル[3]は，WEB アプリの入出力をモデル化し，各種の著名な攻撃が狙う攻撃対象（資産）と攻撃経路を，アプリ開発者に分かりやすく可視化している．一方，HTML5 アプリでは，クライアント上に存在する攻撃対象が多様化し，かつクライアントへの攻撃経路が多様化するため，セキュリティモデル（図1）は，クライアント（図中「攻撃対象端末」）を中心にモデル化を行った．

具体的には，攻撃対象（クライアント上攻撃対象①～③，サーバ上攻撃対象④～⑤），攻撃経路（クライアントーサーバ間攻撃経路①～②，クライアント内攻撃経路③，クライアント間攻撃経路④）を想定して，モデル化を行った．

3　セキュリティリスクの分類体系

本研究では，セキュリティモデル（図1）に基づき，既存のセキュリティレポートより抽出したセキュリティリスクの分類に適した分類体系を構築した（図2）．本分類体系の「大分類」は，アプリ開発者に「何を何故守るのか」というセキュリティ観点を伝えるために設けており，セキュリティモデル（図1）中の「攻撃対象」「攻撃経路」に対応する．本分類体系の「中分類」は，アプリ開発者が対策の必要性を判断するために設けており，HTML5 の各 API に対応する．アプリ開発者は，利用する API に対応する中分類を把握し，その中分類下の対策を行う必要がある．以降，各大分類の概要と，各大分類に属する中分類（API）について記述する．

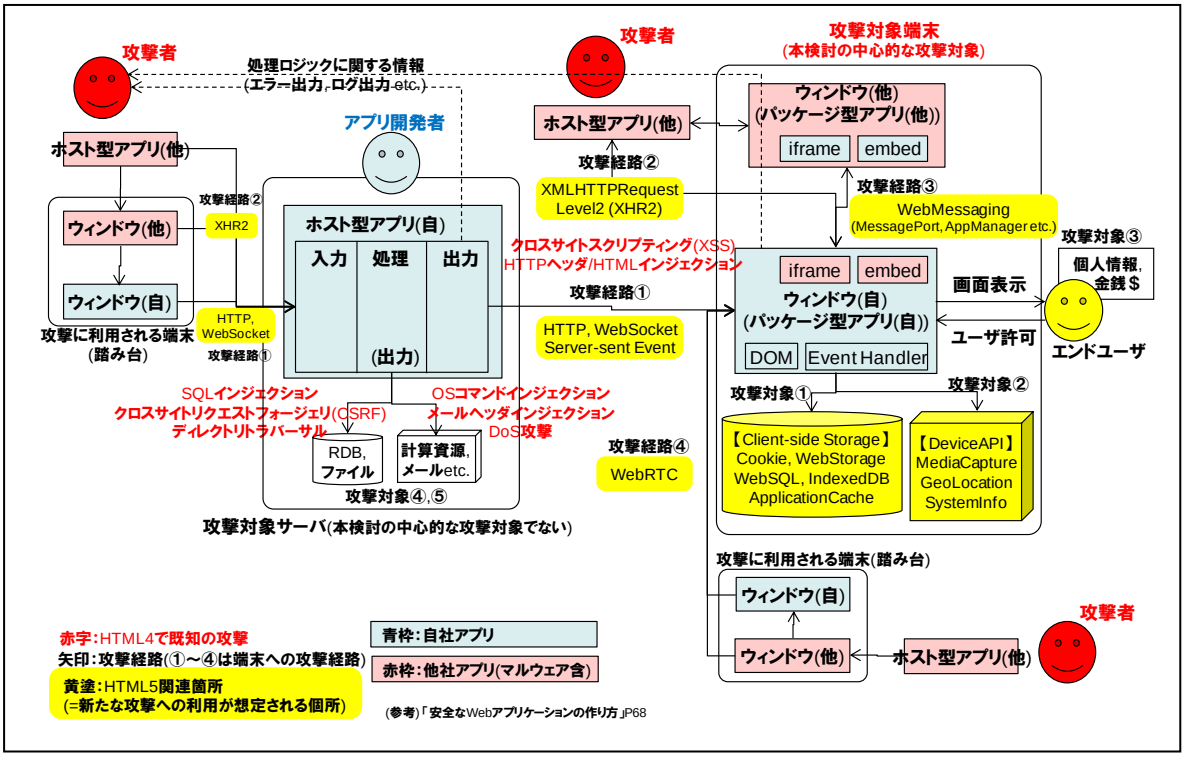

図1：HTML5 アプリケーションのセキュリティモデル（攻撃対象と攻撃経路）

図2:HTML5 アプリケーションのセキュリティリスクの分類体系

大分類1:(クライアント)データの保護

攻撃者は，クライアント上のデータを不正に取得することで，アカウントのなりすまし及び WEB サービスの不正利用，クライアント端末上のクレジットカード情報等の不正利用が可能となる.

クライアント側にデータを保持する API (1-1.WebStorage,1-2.WebSQL,1-3.Indexed DB,1-4.FileAPI,1-5.ApplicationCache) については，大分類1でリスク及び対策を記述する.

大分類2:(クライアント)機能の不正利用防止

攻撃者は，クライアント上の機能を不正利用することで，クライアントの動作の強制停止，位置情報の追跡やカメラ・マイクを用いた盗撮・盗聴によるプライバシーの侵害が可能となる. また，スマートフォン上のアプリであれば，SMS の不正送信による過大請求等が可能となる[6].

本論文では，2-1.WebWorkers, 2-2.GeoLocationAPI,2-3.MediaCaptureAPI ,2-4.SystemInformationAPI を対象とするが，今後，各種 DeviceAPI の仕様策定進行に伴い，大分類2に該当するリスクは多様化すると考えられる.

大分類3:(クライアント)不正な画面出力の防止

攻撃者は，不正な iframe や埋め込みコンテンツを，正規のコンテンツに埋め込むことで，正規のサービス利用者のブラウザを不正に操作し，フィッシングサイトへの誘導や，他サーバへの攻撃の踏み台に利用することが可能となる.

iframe や埋め込みコンテンツ,マークアップを含む，画面出力一般に関連するリスク及び対策は，大分類3に記述する.

大分類4:不正な通信の防止

攻撃者は，HTML5 で導入されたクロスオリジン通信の仕組みや，ウィンドウ間通信，サーバプッシュ通信，リアルタイム通信，双方向通信，クライアント間(P2P)通信といった仕組みを利用して，従来の WEB アプリでは想定されなかった経路より，攻撃の実行が可能となる. また，スマートフォン上であれば，アプリ間通信も攻撃に利用可能となる[6].

通信に関連する API については，(4-1.XMLHTTPRequestLevel2, 4-2.WebMessaging,4-3.Server-SentEvents, 4-4.WebSocket,4-5.WebRTC) 大分類4でリスク及び対策を記述する.

大分類5:(サーバ)データの保護

攻撃者は，サーバ上のデータに不正にアクセスすることで，WEB サイトを改ざんして閲覧ユーザ PC にマルウェアをインストールさせること，サーバ上のクレジットカード番号等個人情報や，政府・企業の機密情報を取得することが可能となる．近年増加している標的型攻撃の目的の多くは，本分類に該当する．

但し，HTML5 関連 API は，一部を除き，クライアント上の攻撃対象の増加，攻撃経路の拡大に寄与する傾向にあるため，本分類については，一般の WEB アプリケーション(HTML4)同様のセキュリティ対策を行うことが重要である[3-5]．

大分類6:(サーバ)機能の不正利用防止

攻撃者は，サーバ上の機能に不正にアクセスすることで，攻撃対象サーバを踏み台とするスパムメールの送信や，WEB サービス/サーバの強制的な停止が可能となる．

但し，HTML5 関連 API は，一部を除き，クライアント上の攻撃対象の増加，攻撃経路の拡大に寄与する傾向にあるため，本分類については，一般の WEB アプリケーション(HTML4)同様のセキュリティ対策を行うことが重要である[3-5]．

大分類7:(サーバ)不正なレスポンス出力の防止

攻撃者は，WEB サイト改ざんやスクリプト埋め込みにより，攻撃対象サーバから攻撃者が意図する不正なレスポンスを出力させることで，サービス利用者のブラウザを不正に操作し，攻撃者サーバと通信させることや，DDoS 攻撃へ加担させることが可能となる．

本分類については，一般の WEB アプリケーション(HTML4)同様のセキュリティ対策を行うことが重要である[3-5]．

大分類8:ロジックの保護

攻撃者は，最終的な攻撃対象である個人情報，機密情報等にアクセスするための準備として，攻撃対象アプリケーションの処理ロジックの解析を行うことで，不正アクセスに利用可能な攻撃経路を検出することが可能となる．

一般にアプリケーション開発者は，処理ロジックを保護するために，ソースコード難読化等の対策を行うが，WEB アプリの場合，クライアントに送信された Javascript コードに含まれるロジックの保護は容易ではない．スマートフォン上であれば，OS が提供するパッケージ化機能を利用して対策が可能である[6]．

次章では，上記 8 分類のうち，主に HTML5 に関連する大分類 1～4 に属する API について，各 API が抱えるリスクの傾向と対策を記述する．

4　各 API のリスク抽出結果

◇リスク抽出方法

本研究では，HTML5 関連の各 API について，現時点で想定しうるリスクを抽出し，リスクの傾向を分析し，各リスクへの対策を立案した．リスクの抽出においては，ENISA[1],OWASP[2]が作成したセキュリティレポートを参考とした．各レポートの傾向は以下である．

ENISA[1]:概念的な記述が多く，リスクが顕在化するシーンの想定が難しい項目，リスクに対する対策の立案が難しい項目が含まれる．尚，OWASP で解説済みのリスクについては記載していない．

OWASP[2]:実装の観点の記述が多く，実践が可能な対策が多い．リスクが顕在化するシーンが想定しやすい項目が多い．

上記 2 点のレポートより，リスク抽出を行い，類似するリスク・対策の整理を行った結果，計 76 項目のチェックリストとなった．リスクの種別としては，脆弱性に直接関連する項目，プライバシーポリシー等の事業方針に依存する項目，アプリの機能性に関する項目，ブラウザ開発者のレベルで対策がなされるべき項目があった．

◇大分類1:(クライアント)データの保護

・中分類 1-1:WebStorage

WebStorage には,
LocalStorage と SessionStorage があり, データの公開範囲や保持期間が異なるため, 機能要件・セキュリティ要件等を考慮して, 適切に使い分けを行う必要がある.

LocalStorage は, 同一生成元ポリシーが適用される永続的ストレージである. サブドメイン毎にデータアクセスを制限でき, サブドメイン毎に別々のアプリケーションを動作させられる. これに対して, Cookie はパス毎にデータアクセスを制限するため, LocalStorage とは挙動が異なる. 左記の点を考慮せず, Cookie の代替として LocalStorage を使用していた場合, アプリケーション間でデータ衝突を起こす可能性がある.

SessionStorage は, データを利用するウィンドウが無くなると同時に消去される一時的ストレージである. 一時的に保管すべきデータは, LocalStorage に保管していた場合, ブラウザ終了後もデータが保持されて攻撃を受けるリスクが高まるため, SessionStorage に保管すべきである.

LocalStorage,SessionStorage 共に, Javascript アクセスが前提であり, クロスサイトスクリプティング脆弱性が存在する場合には, データ保護は保証されない. そのため, セッション ID 等の重要データは, 「httpOnly」フラグを設定した Cookie に保管すべきである. また, クライアント側に保持するデータは最小限にして, 重要データはサーバ側に保管することを検討すべきである.

・中分類 1-2:WebSQL

WebSQL は, W3C が 2010 年 11 月に非推奨としており, ブラウザベンダのサポートが止まる可能性があるため, 脆弱性パッチが適用されなくなる, 機能性が損なわれる等のリスクがある. 代替として, IndexedDB を使用すべきである.

アプリケーション互換性等の問題で, WebSQL を使用せざるを得ない場合には, クライアント上の SQL インジェクションに対して, 従来の WEB アプリ同様の対策が必要である. また, クライアント側に保持するデータは最小限にして, 重要データはサーバ側に保管することを検討すべきである.

・中分類 1-3:IndexedDB

・中分類 1-4:FileAPI

本論文の執筆時点で, IndexedDB, FileAPI に関する体系的なセキュリティレポートが存在しないため, 本論文では詳細に言及しない. アクセス権限の管理等, 仕様を理解した上で利用することが適切である.

・中分類 1-5:ApplicationCache

アクセス制限が設定されていない公衆無線 LAN 等, 安全でないネットワーク上で WEB アプリを使用する場合, 攻撃者によりアプリケーションキャッシュが改ざんされ, 不正なコードが実行される可能性がある. このため, アプリケーションキャッシュを利用する際は, なるべく SSL/TLS 等でサーバ認証を行うことが望ましい.

◇大分類2:(クライアント)機能の不正利用防止

・中分類 2-1:WebWorkers

WebWorkers は, バックグラウンドで Javascript 実行を可能とする API である. バックグラウンドで実行される点を悪用して, 不正なスクリプトを WebWorkers に渡し, クライアントの CPU 負荷を上げる, 一種の DoS 攻撃が行われる可能性がある. このため, WebWorkers に渡すスクリプトには, 不正なコードが含まれないか確認する必要がある.

また, WebWorkers は, クロスオリジンのリクエストを行うことが可能であるため, 中分類 4-1 のリスクに対する対策が必要である.

・中分類 2-2:Geo-LocationAPI

Geo-LocationAPI は, 直接脆弱性を作り込

む可能性は低いが，プライバシーポリシーの範囲を越えて位置情報を取得してしまうリスクがある．位置情報の取得時は，ユーザ許可を適切なタイミングで要求すべきである．「適切なタイミング」は，各事業者のプライバシーポリシーに依存する[7]．例えば，位置情報の取得は一回限りの取得か常時監視か，位置情報はキャッシュ可能か，位置情報の取得許可はキャッシュ可能か，等といった点に関して，ポリシーを策定してユーザに提示する必要がある．

・中分類 2-3：MediaCaptureAPI
・中分類 2-4：SystemInformationAPI

Geo-LocationAPI と同様に，プライバシー情報の取得に相当するため，プライバシーポリシーの範囲を越えて情報を取得してしまうリスクがある．ユーザ許可を適切なタイミングで要求すべきである[7]．

◇大分類3：不正な画面出力の防止

・中分類 3-1：iframe 関連

iframe 内で他サイトのコンテンツを表示する場合，iframe 内の不正なコンテンツ経由で情報が漏えいする可能性があるため，iframe の sandbox 属性を設定する必要がある．sandbox 属性により，iframe 内のスクリプト実行やフォーム送信を制限でき，iframe 内に不正なコンテンツが読み込まれた場合に，不正な動作を抑制できる．但し，sandbox 属性ではプライバシー情報の取得を制限できないため，iframe 内ではプライバシー情報取得 API へのアクセスを別途制限すべきである．

一方，自サイトのコンテンツが外部サイトの iframe 内に表示される場合には，自サイトのコンテンツがクリックジャッキング攻撃に利用される可能性がある．X-Frame-Options ヘッダを使用して，外部参照を許可するサイトをホワイトリストで指定すべきである．

その他，画面出力に関連するリスクとして，HTML5 では HTML タグでフォーム送信等のアクションが可能であるため，HTML タグのインジェクションにより，不正な処理が実行される可能性がある．従来のスクリプトインジェクション対策に加えて，HTML タグのインジェクション対策が必要である．

◇大分類4：不正な通信の防止

・中分類 4-1：XMLHTTPRequestLevel2

XMLHTTPRequestLevel2(XHR2)は，クロスオリジンのリクエストを可能とする API である．クロスオリジンのリクエストを受信するサーバにおいて，サーバ上のリソースを保護するために，HTTP リクエストの「Origin」ヘッダと，レスポンスの「Access-Control-Allow-****」ヘッダを利用したアクセス制御の機構が提案されている．しかし，非ブラウザのリクエストに対しては左記の機構が機能しない可能性があり，適切なアクセス制御が行われず，サーバ上の情報が漏えいする可能性がある．このため，上記機構とは別に，アプリケーション側でアクセス制御を行うべきである．その上で，上記機構を，リスク軽減の観点で利用すべきである．

「Access-Control-Allow-****」ヘッダによるアクセス制御については，
「Access-Control-Allow-Origin:*」というワイルドカード指定や，リクエストの「Origin」ヘッダを「Access-Control-Allow-Origin」でエコーバックする設定では，レスポンスに含まれる情報が漏えいする可能性がある．このため，レスポンスに機密情報や攻撃者に有益な情報が含まれる場合には，「Access-Control-Allow-Origin」でレスポンス受信を許可するオリジンをホワイトリストで指定すべきである．

・中分類 4-2：WebMessaging

WebMessaging は，ウィンドウ間通信を行うための API である．

メッセージ送信側では，メッセージ送信先として「*(ワイルドカード)」を指定していた場合，意図しないウィンドウにメッセージを受信され，情報が漏えいする可能性があるため，メッセージ送信先を FQDN で明示的に指定すべきである．

メッセージ受信側では，意図しないウィンドウからのメッセージを受信する場合，インジェクション等の攻撃が行われる可能性があるため，送信元が意図したウィンドウであるかを FQDN で確認すべきである.

・中分類 4-3：Server-sentEvents

Server-sentEvents は，サーバプッシュ通信を行うための API である.

クライアントが，意図しないサーバからプッシュされたメッセージを受信する場合，インジェクション等の攻撃が行われる可能性があるため，送信元が意図したサーバであるかを FQDN で確認すべきである.

・中分類 4-4：WebSocket

WebSocket は，リアルタイム通信・双方向通信を行うための API 及び通信プロトコルである.

WebSocket プロトコルは，アクセス制御の機構を持たず，通信内容が漏えいする可能性があるため，アプリケーション側でアクセス制御を行うべきである.

また, SSL/TLS を使用しない場合，中間者攻撃を受けて通信内容が漏えいする可能性があるため，wss://(WebSocket over SSL/TLS)を使用すべきである.

WebSocket サーバは，WEB サーバ同様に，SQL インジェクション対策を行うべきである.

・中分類 4-5：WebRTC

本論文の執筆時点で，WebRTC に関する体系的なセキュリティレポートが存在しないため，本論文では詳細に言及しない．通信先の指定方法等，仕様を理解した上で利用することが適切である.

5　各 API のリスクの傾向

ENISA,OWASP より抽出したリスク項目には，実際にはセキュリティリスクでないリスク項目が多く含まれることが分かった．そこで，リスク種別として以下4種別（「脆弱性」「ポリシー」「機能性」「その他」）を設け，種別毎に集計を行い，各 API のリスクの傾向を分析した（表1）．ここで，「脆弱性」はセキュリティリスクに関する種別，「ポリシー」はプライバシーポリシーやサービス規約等，リスクの扱いが事業方針に依存する種別，「機能性」はアプリケーションの機能性に関する種別，「その他」は主にブラウザ開発者向けのリスクに関する種別を示す.

大分類1,大分類4に属する API については，「脆弱性」に直接関連する項目が多かった．これは，データアクセスや通信について，既に脆弱性が生じうる個所が予測されており，アプリ開発者による対策が必要であることを示す．特に，HTML5 特有のストレージに不正なコードを挿入されるリスク，HTML5 特有の通信 API で不正なコードを送信されるリスクは明らかであり，ストレージ関連及び通信関連の API を使用する際は，4章で述べた各対策に加えて，リスク軽減の観点で入力値検証が必要である.

大分類2に属する API については，「ポリシー」に分類される項目が多かった．これらAPIを使用する際，アプリ開発者は，自社のプライバシーポリシーに則って適切なタイミングでプライバシー情報取得許可をユーザに要求すべきである[7].

大分類3に属する画面出力関連の API については，ブラウザが規定する動作に基づくリスクが多いために，「その他」の項目が増えている．これは，ブラウザ側で対策がなされるべき項目であり，アプリ開発者が対策を行う必要は無い．そのため，他の「脆弱性」「ポリシー」「機能性」のリスクに対する対策がより重要である.

6　まとめ

本研究では，既存の HTML5 セキュリティレポート(ENISA,OWASP)を分析し，アプリ開発者が対策すべきリスク（計 76 項目）を抽出した．このうち本論文では，リスク抽出の過程で得られた，各 API が抱えるリスクの傾向，リスクの分類体系，及びセキュリティモデルについて記述した.

本論文の段階では，仕様策定が進行中のAPI，及び仕様策定は完了しているがセキュリティレポートが存在しない API について，分析が不十分である．これら API については，今後の仕様策定の進行，及びセキュリティレポートの発行に合わせて，改めてセキュリティリスクの抽出・整理を行うべきと考えている．

また，HTML5 アプリケーションをネイティブアプリ相当の権限で動作可能とするスマートフォン OS が今後増加すると考えられる．これら OS 上で HTML5 アプリを動作させる場合には，本論文で述べた観点に加えて，各 OS 固有の API が有するリスクと，各 OS 固有のセキュリティ機能の有効性について，新たに検討が必要である．合わせて，マルウェアのインストールに対する対策が重要である．

[1] European Network and Information Security Agency (ENISA), “A Security Analysis of Next Generation Web Standards”,2011/7/31,

[2] Open Web Application Security Project (OWASP), “HTML5 Security Cheat Sheet”, 2013/04/01(WEB ページ参照時)

[3] 徳丸 浩, ”体系的に学ぶ 安全な Web アプリケーションの作り方”, 2011/03/03,

[4] Michal Zalewski, ” The Tangled Web: A Guide to Securing Modern Web Applications”, 2011/11/22 ,

[5] IPA, “安全なウェブサイトの作り方 セキュリティ実装 チェックリスト”, 2012/12/26,

[6] JSSEC, “Android アプリのセキュア設計・セキュアコーディングガイド”, 2013/04/23,

[7] 総務省, “スマートフォン プライバシー イニシアティブ”, 2012/08/07,

表1：HTML5 の API が有するリスク（ENISA/OWASP 記載）の傾向

	脆弱性	ポリシー	機能性	その他	
1-1.Web Storage	**6**		2	1	
1-2.(Web SQL Database)	**4**				
1-3.Indexed Database API					
1-4.File API					
1-5.Application Cache	1	2	1		
2-1.Web Workers	**3**				
2-2.Geo-Location API		**7**	1		
2-3.Media Capture API		**2**			
2-4.System Information API		**4**	1		
3-1.iframe 関連	3	3	3	**6**	
4-1.XMLHTTPRequest Level2	**8**		1	5	
4-2.Web Messaging	**4**			1	
4-3.Server-Sent Events	**2**				
4-4.Web Socket	**5**				
4-5.Web RTC					
計	36	18	9	13	76