Vol.2015-CSEC-69 No.20
Vol.2015-IOT-29 No.20
2015/5/22

Firefox拡張機能が実行しうる操作の通知システム

高橋　研介[1]　高橋　一志[1]　大山　恵弘[1]

概要：近年，代表的なWebブラウザの一つであるFirefoxの拡張機能を悪用した攻撃の危険性が認識されており，悪意を持つ拡張機能による脅威が増している．一方，Firefoxでは拡張機能をインストールする際，その拡張機能がどのような操作を実行するのか一切通知されない．そのため，無害に見える拡張機能がバックグラウンドで悪質な操作を行っている場合，それを発見することは極めて難しい．本稿では，そのような状況への対策として，Firefox拡張機能が実行しうる操作をユーザに通知するシステムの提案を行い，インストールする拡張機能が悪意を持つかどうかを判断する材料を提供する．また，セキュリティ対策の一環として2015年に導入される予定であるFirefox拡張機能の署名が，本システムに与える影響を考察する．

Notification System of Operations that May Be Executed by Firefox Extensions

Kensuke Takahashi[1]　Kazushi Takahashi[1]　Yoshihiro Oyama[1]

Abstract: Risks of attacks that exploit Firefox extensions have been recognized recently, and threats of malicious Firefox extensions are widely spread. On the other hand, when a Firefox extension is installed, Firefox does not notify users about operations executed by the extension. Thus, if an extension executes malicious operations in the background, it is very difficult to detect them. In this paper, we propose a system for notifying users of operations that may be executed by Firefox extensions as a countermeasure to such a situation. This system provides information for making a decision about whether the Firefox extension is malicious. Furthermore, we consider the effect on this system produced by extension signing, which is going to be introduced into Firefox as a part of the security measures in 2015.

1. はじめに

Webブラウザにおける拡張機能とはWebブラウザに新たな機能を追加したり，Webブラウザのデザインを変更したりするための小さなサブプログラムである．Mozillaにより開発が行われているFirefoxでは，多くの拡張機能が使用されている．拡張機能によって新たな機能を追加できることは，Firefoxを使用する上で大きなメリットである．ところが近年，Firefox拡張機能を悪用した攻撃の危険性が認識されており，悪意を持つ拡張機能による脅威が増している[1]．さらに，Mozillaにより公表されている拡張機能のブロックリスト[2]に登録される拡張機能は，**図 1** に見られるように直近3年間で急激にその数を増やしている．ブロックリストに登録されている拡張機能は，Firefoxの安全性や安定性に関して問題を引き起こす悪質な拡張機能である．このことからも悪意を持つ拡張機能による脅威が増していることが伺える．

その一方，Firefoxでは拡張機能をインストールする際，その拡張機能がどのような操作を実行するのか一切通知されない．そのため，無害に見える拡張機能がバックグラウンドで悪意の挙動を行っている場合，それを発見することは極めて難しい．例えば，ツールバーを装っているが，バックグラウンドでは重要なファイルを削除したり，Firefoxとは別のプロセスを実行したりといった悪質な拡張機能である．こういった悪質な拡張機能による脅威は日々増しており，その対策が求められている．

本稿では，そのような状況への対策として，Firefox拡張機能が実行しうる操作をユーザに通知するシステムの

[1] 電気通信大学
The University of Electro-Communications

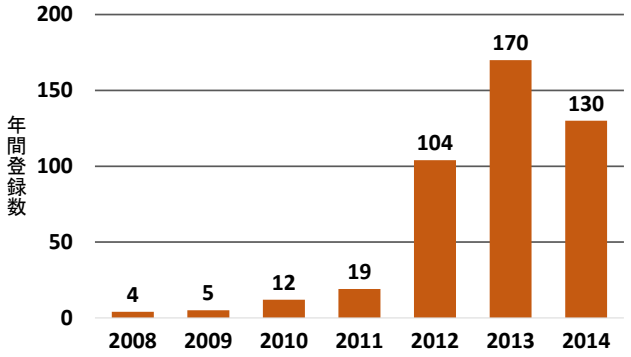

図 1　ブロックされた拡張機能の年間登録数 ([2] を基に作成)

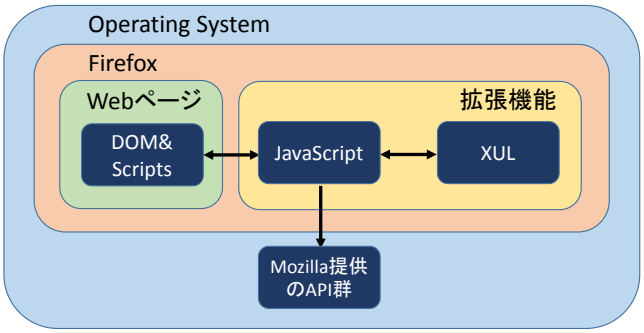

図 2　Firefox 拡張機能の構成 ([3] より一部改変)

提案を行う．通知システムは拡張機能のインストールの際に操作の通知を行い，通知後，ユーザは拡張機能のインストールの可否を決定することができる．通知システムは拡張機能のソースコードに含まれるスクリプト内に記述されている，悪質な挙動に繋がる操作に関するメソッドやプロパティの使用を静的解析により検出する．そして，インストールする拡張機能が悪意を持つかどうかを判断する材料を通知システムにより提供する．

本稿は全 7 章から構成されている．2 章では Firefox 拡張機能の構成や，Firefox 拡張機能の開発に用いる API 群について説明する．3 章では既存研究を紹介し，通知システムが解決すべき問題点について述べる．4 章では通知システムの設計と実装について説明し，5 章では通知システムを使用した実験を行う．6 章ではセキュリティ対策の一環として 2015 年に導入される予定である Firefox の署名の概要や，本稿で提案する通知システムへの影響について述べる．最後に 7 章では本稿をまとめ，今後の課題について述べる．

2. Firefox 拡張機能

本章では，Firefox 拡張機能の構成と，本稿で扱う Firefox 拡張機能を開発するために Mozilla が提供している 4 種類の API 群について説明する．

2.1 Firefox 拡張機能の構成

Firefox 拡張機能の構成を**図 2** に示す．Firefox 拡張機能は主に 2 種類の言語から記述される．一つは，XUL という Mozilla アプリケーション用に開発された言語で，Firefox の UI 部分を担当している．もう一つは，JavaScript で，Firefox 内部の動作制御部分を担当している．それらに加え，以降で説明する Firefox 拡張機能を開発するために Mozilla が提供している 4 種類の API 群を使用することで，XUL や JavaScript だけでは実現不可能な，Firefox 本体や OS に対する高度な処理を行うことが可能となる．しかし，高度な処理を行うことが可能となる一方で，これらは悪質な挙動に繋がる操作の要因となっている．

2.2 Mozilla が提供する API 群

Firefox 拡張機能を開発するために Mozilla が提供している 4 種類の API 群について説明する．

XPCOM コンポーネント（XPCOM）

XPCOM コンポーネントはプラットフォームに依存せずに，XUL や JavaScript だけでは実現不可能な，Firefox 本体や OS に対する高度な処理を行うことを可能にする．XPCOM コンポーネント自体は C++で記述されているが，XPConnect と呼ばれる技術を用いることで，JavaScript から XPCOM コンポーネントを呼び出すことができる．また，XPCOM コンポーネントは非常に多くの機能を備えている一方，攻撃に悪用される恐れが指摘されている [4]．

FUEL

FUEL は XPCOM に比べ，コードの記述方法や API を JavaScript に近づけることで，拡張機能の開発をより行い易くするための技術である．後述の JavaScript コードモジュールや Add-on SDK モジュールの登場でほとんど使用されることはなくなったが，機能の廃止には至らず，Firefox 拡張機能から FUEL を使用することができる．

JavaScript コードモジュール

JavaScript コードモジュールは，再利用性のある JavaScript コードをインポート可能な形でモジュール化したものである．多くの標準コードモジュールが Firefox に組み込まれており，拡張機能に標準コードモジュールのファイルを含めずに拡張機能から呼び出すことができる．

Add-on SDK モジュール

Add-on SDK とは 2010 年に新たに登場した Firefox 拡張機能の開発環境であり，これまでの開発手法における XPCOM や XUL といった Mozilla アプリケーション固有の技術を必要とせずに，拡張機能の開発を行うことができる．Add-on SDK モジュールとは，Add-on SDK による拡張機能の開発において使用される専用の API 群である．

3. 既存研究

本章では，Firefox 拡張機能が実行する操作の通知や，Firefox 拡張機能の解析を目的とした既存研究を紹介し，通知システムが解決すべき既存研究の問題点について述べる．

3.1 Firefox 拡張機能の操作の通知を目的とした研究

Marston ら [5] は，Android 版 Firefox において，Firefox 拡張機能のインストール時，もしくは更新時に静的解析を行い，ユーザ情報を外部に送信する操作を検出した場合，ユーザに通知するシステムを実装している．システムは Android 版 Firefox 本体の改良と，Firefox 拡張機能の両方を組み合わせて実装されている．静的解析ではソースコードに対してフロー解析を行い，ユーザ情報が外部に送信されるかどうかを検証する．このシステムにおける問題点として，ユーザ情報を外部に送信する操作以外の悪意の操作を検出できない点が挙げられる．様々な悪意の操作の検出に対応していないことが問題点となるのは，Mozilla により提供される API 群による悪意の操作は非常に多岐に渡るからである．例として，Abraham [6] は Firefox 拡張機能によってキー入力情報の盗聴，任意のプロセスの実行，Firefox 上のセッション情報の盗聴，Linux のパスワードファイルの盗聴，DDoS 攻撃といった様々な攻撃を検討している．そのため，様々な悪意の操作に対応するセキュリティシステムが求められる．また，Firefox 本体の改良による実装では，ユーザに対する導入の容易さや，Firefox の更新に逐一対応することが難しいという問題点が存在する．

3.2 Firefox 拡張機能の解析を目的とした研究

以下に紹介する研究は Firefox 拡張機能の解析を目的とした研究だが，ユーザに対する操作の通知が目的ではない．

Beacon [4] は，WALA という既存の静的解析システムを改良し，Add-on SDK の前身である Jetpack を使用して開発された Firefox 拡張機能や，Jetpack モジュールを静的解析できるようにしたシステムである．静的解析では，拡張機能のソースコードをコールグラフ化し，XPCOM の使用や，モジュール外へのデータフローを解析している．問題点として，解析におけるオーバーヘッドとして 1 モジュールあたり平均数分，最大で 30 分を要しており，このシステムをユーザに対する操作の通知へ応用した場合，オーバーヘッドが大きすぎる点が挙げられる．

VEX [7] は Firefox 拡張機能内のスクリプトの情報フローを静的解析することで，不正な eval() の使用や，DOM の改ざんを発見するシステムである．問題点として，不正な eval() の使用や，DOM の改ざん以外の悪意の操作を検出できない点が挙げられる．

Shahriar ら [8] は，拡張機能のソースコードに含まれている JavaScript と XPCOM の API 名を基に，悪意を持たない拡張機能，脆弱な拡張機能，悪意を持つ拡張機能にそれぞれモデル化することで，脆弱な拡張機能や悪意を持つ拡張機能を検出するシステムを実装している．問題点として，XPCOM 以外の API 群を解析対象としていないため，それらの API 群を使用して悪意の操作を行うことで解析の回避が可能である点が挙げられる．Firefox 拡張機能の静的解析では，様々な API 群を解析対象にする必要がある．

Wang ら [9] は Firefox 拡張機能が実行する操作を自動でテストすることで，動的解析を行うシステムを Firefox に組み込むことで実装している．動的解析では，Firefox 拡張機能の操作を 4 段階に危険度別でレーティングし，JavaScript や XPCOM，XPConnect の使用を監視することで，レーティングされた該当の操作の実行を検証する．動的解析の問題点として，特定の Web サイトへのアクセスや，特定のユーザアクションをトリガとして操作が実行される場合は，その操作を検出できない点が挙げられる．

4. 通知システムの設計と実装

本章では，提案する通知システムの設計と実装について説明する．また，実装における静的解析の手法や，スクリプトの動的実行対策について述べる．

4.1 通知システムの設計

通知システムでは拡張機能のソースコードに含まれるスクリプト内に記述されている悪質な挙動に繋がる操作に関するメソッドやプロパティの使用を静的解析により検出する．拡張機能のインストールが行われようとすると，その拡張機能が実行しうる操作の一覧がブラウザ上に通知され，ユーザは拡張機能のインストールの可否を決定することができる．

通知システムは，以下の流れで動作を行う．

(1) 拡張機能のインストール処理が開始される．
(2) 通知システムが拡張機能のインストール処理を中断する．
(3) ネットワーク上に存在する拡張機能をインストールしようとしていた場合，通知システムは拡張機能の本体を一時フォルダにダウンロードする．ローカル上に存在する拡張機能をインストールしようとしていた場合，通知システムは拡張機能の本体を一時フォルダにコピーする．
(4) 通知システムが拡張機能のソースコードに含まれるスクリプトを静的解析し，拡張機能が実行しうる操作を記録する．
(5) 通知システムがブラウザ上に解析結果を通知する．
(6) 拡張機能のインストールの可否をユーザが選択する．
(7) 「はい」ならば，通知システムにより拡張機能がインストールされる．
(8) 「いいえ」ならば，拡張機能はインストールされない．

4.2 通知システムの実装

通知システムはユーザに対する導入の容易さや，Firefox の高速リリースによるアップデートへの対応を考慮し，Firefox 拡張機能により実装を行う．なお，Firefox 拡張機能では，他の拡張機能に対して，インストールの中断や削除といった操作を実行することが可能である．そのため，通知システムをインストールする以前に，悪意を持つ拡張機能がインストールされている場合，通知システムのインストールを妨害される恐れがある．そこで，本稿では通知システムをインストールする際，Firefox に悪意を持つ拡張機能がインストールされていないことを前提とする．

通知システムによる拡張機能のインストールの中断や，インストールといった，拡張機能の管理には標準コードモジュールである AddonManager.jsm を使用する．まず，Firefox が起動すると，AddonManager.jsm により拡張機能のインストールに対するリスナが登録される．拡張機能のインストールが行われようとするとそのインストールは中断され，静的解析の処理が始まる．解析結果の通知ダイアログにおいて，「はい」のボタンが選択されると，AddonManager.jsm により，ダウンロードまたはコピーされた拡張機能本体が使用されインストールが行われる．

インストール処理の中断後，拡張機能本体を入力として静的解析を行うが，その前に必要となる事前処理について述べる．Firefox 拡張機能のフォーマットである xpi 形式は，拡張機能の要素を zip 形式でアーカイブしたファイルである．そこで，事前処理ではまず xpi ファイルを XPCOM の nsIZipReader を用いて展開する．続いて，展開されたエントリのファイル形式を拡張子によって判断し，JavaScript ファイル（.js），JavaScript コードモジュールファイル（.jsm），html ファイル（.html）を静的解析の対象とする．JavaScript コードモジュールファイルは JavaScript で記述されているため，html ファイルは JavaScript が含まれる場合が存在するためである．また，一部の拡張機能には xpi 形式でアーカイブする前に，jar 形式で二重にアーカイブしているものが存在するため，展開されたエントリが jar ファイルである場合，再帰的に事前処理と静的解析を行う．

静的解析が終了すると，解析結果が**図 3** で示したように，XPCOM の nsIPromptService を使用したダイアログ形式でブラウザ上に通知される．下部にボタンを設置することで，ユーザが拡張機能をインストールするかどうかを選択することができる．通知内容は拡張機能の名称とインストール元の URI に加え，操作の内容と，独自に定義したその操作が悪用された場合の危険度，その操作が悪用された場合の脅威の 3 項目とした．また，通知された操作は悪用されていない場合があるため，拡張機能の説明文と比較することを推奨する注意書きを併記する．

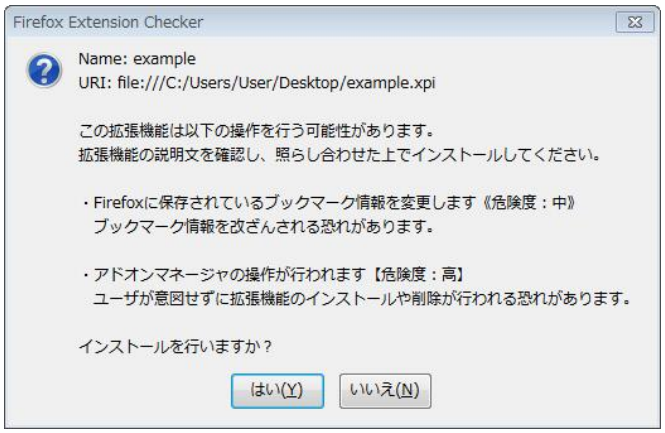

図 3 通知ダイアログの例

表 1 検出対象の操作の一覧

カテゴリ	操作	危険度
個人情報	ログイン情報の読み取り	高
	クッキー情報の読み取り	高
	クッキー情報の変更	中
	位置情報の取得	高
Firefox 内部情報	Firefox の設定の変更	中
	ブックマーク情報の読み取り	中
	ブックマーク情報の変更	中
	閲覧履歴の読み取り	低
Firefox 全般	アドオンマネージャの操作	高
	Firefox の UI の変更	低
	Firefox の強制終了	低
	アイドル状態の取得	低
ネットワーク	ネットワーク通信	中
	Wi-Fi 状態の取得	中
ストレージ	ファイルのダウンロード	高
	ファイルの削除	中
	ファイルのオープン	中
システム	メモリの確保	中
	プロセスの実行	高
	クリップボードの読み取り	高
	クリップボードへの設定	中

4.3 静的解析

静的解析では拡張機能のソースコードに含まれているスクリプト内において，悪質な挙動に繋がる操作を実行することができるメソッド名やプロパティ名を，JavaScript の indexOf() を使用した文字列検索により検出する．**表 1** は検出の対象とした悪質な挙動に繋がる操作の一覧である．「個人情報」，「Firefox 内部情報」，「Firefox 全般」，「ネットワーク」，「ストレージ」，「システム」の 6 カテゴリ，合計 21 個の操作を検出の対象とした．右列は解析結果の通知に用いられる，独自に定義した操作が悪用された場合の危険度であり，3 段階で定義している．

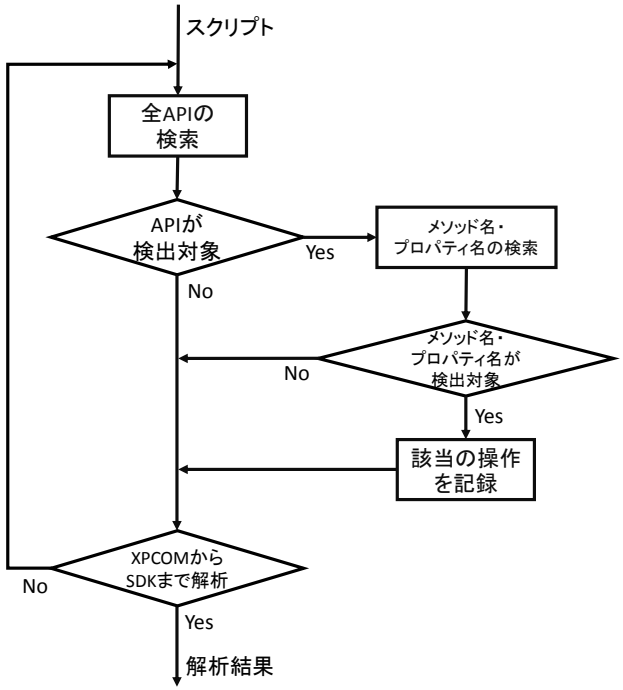

図 4 静的解析のフロー

次に**図 4** に示した静的解析の手順について述べる．まずスクリプト内に存在する XPCOM と FUEL の全コントラクト ID を文字列検索により解析する．コントラクト ID とは XPCOM や FUEL の API に対応する ID であり，API の呼び出しの際に参照される．そして，コントラクト ID により解析した API の中に検出対象の API が含まれていた場合，次はその API が持つ検出対象のメソッド名やプロパティ名で文字列検索を行う．もしそのメソッド名やプロパティ名がスクリプト内に含まれていた場合，該当の操作を実行しうる操作として記録する．これを同様に JavaScript コードモジュール，Add-on SDK モジュールに対して順に行う．なお，JavaScript コードモジュール，Add-on SDK モジュールにおける API の呼び出しはコントラクト ID ではなく，API 名による呼び出しのため，コントラクト ID の代わりに API 名による文字列検索により解析を行う．最終的に，拡張機能内に含まれる全ての解析対象ファイルに対して静的解析が終了すると，解析結果である実行しうる操作の一覧がブラウザ上に出力される．

なお，API を検出した後，メソッド名やプロパティ名による文字列検索を行う現状の手法では，別 API の同一名称のメソッド名やプロパティ名を検出してしまうという誤検出の可能性が存在する．例えば，通知システムでは「ファイルの削除」の操作において，XPCOM である nsIFile の remove() というメソッドの使用を検出の対象としている．このメソッドは別 API である nsICookieManager にも存在し，もし同スクリプト内において，nsIFile と nsICookieManager 両方の API が使用されており，nsICookieManager の remove() が使用されていた場合，nsIFile の remove() として検出され，該当の操作が行われていないにもかかわらず，「ファイルの削除」が通知されてしまう．解決策として考えられるのは，検出対象のメソッドやプロパティを呼び出したインスタンスの生成元が，検出対象の API であるかどうかを検証することであり，誤検出の要因を解決することは今後の課題の一つである．

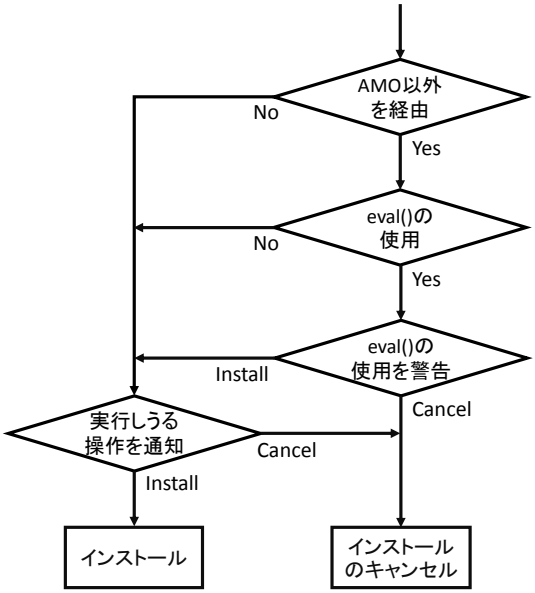

図 5 スクリプトの動的実行対策のフロー

4.4 スクリプトの動的実行対策

JavaScript の静的解析を回避するための代表的な手法の一つが，難読化したスクリプトの eval() による動的実行である．通知システムでは，eval() による動的実行の対策として**図 5** で示したように，Mozilla の公式サイトである addons.mozilla.org （AMO）以外を経由して拡張機能がインストールされる場合，スクリプト内での eval() の使用を静的解析し，使用が検出された場合，警告を表示してインストールをキャンセルできるように実装している．なお，AMO で配布されている拡張機能は eval() の使用が基本的に禁止されており，Mozilla による審査を通過できない [10]．審査を通過している拡張機能内で eval() が使用されている場合，審査によって安全な eval() の使用が検証されているため，AMO で配布されている拡張機能の場合には，eval() の使用による警告表示の対象とはしていない．

5. 実験

本章では，通知システムの有効性を検証するため，実際に拡張機能の解析を行い，解析結果の通知例を紹介する．また，通知システムの導入に対するオーバーヘッドを計測する．実験環境はブラウザが Mozilla Firefox 36.0.1，OS が Microsoft Windows 7 SP1（64bit），RAM が 8GB，CPU が Intel Core i7-2600 3.40GHz である．

5.1 通知例 1

自作した悪意を持つ拡張機能を使用する．この拡張機能はインストール直後に，Firefox に保存されているログイン情報を読み取り，そのログイン情報を外部サーバに送信する機能を備えている．拡張機能の実行例として，ターゲット側から送信されたログイン情報（ホスト名，ID，パスワード）が攻撃者側のブラウザ上に表示された様子を**図 6** に示す．攻撃者側ではサーバサイド JavaScript である Node を利用し，HTTP サーバを立てることで，ターゲット側から送信されたログイン情報をブラウザ上に表示できる．

解析結果の通知例を**図 7** に示す．「ログイン情報の読み取り」はホスト名，ID，パスワードの読み取りに対して通知されたもので，「ネットワーク通信」は読み取った情報の送信に対して通知されたものである．

5.2 通知例 2

AMO で配布されている拡張機能である Classic Theme Restorer（Customize UI）[11] を使用する．この拡張機能は Firefox のデザインを以前のデザインに再現することを目的に開発され，多くの部分の UI をカスタマイズすることが可能である．また，デザインだけでなく，UI に対するアクションを変更することも可能である．

解析結果の通知例を**図 8** に示す．「Firefox の設定の変更」は Firefox のデザインに関する設定や，アクションの設定の変更に対して通知されたもので，「Firefox の UI の変更」は拡張機能内に含まれているスタイルシートの適用によるデザインの変更に対して通知されたものである．「Firefox の強制終了」は拡張機能の一部の設定の変更に伴い，Firefox の再起動が必要となり，その際に一旦 Firefox を終了させることに対して通知されたものである．

5.3 通知例 3

AMO で配布されている拡張機能である IE Tab V2（Enhanced IE Tab）[12] を使用する．この拡張機能は Firefox 中で Internet Explorer のレンダリングエンジンを使用し，Web ページを表示することができる拡張機能であり，Internet Explorer 以外のブラウザでは表示できない Web サイトを Firefox 上で表示させる用途に使用される．

解析結果の通知例を**図 9** に示す．「Firefox の設定の変更」は Web サイト上のプラグインの表示に関する設定の変更や，拡張機能本体の設定の変更に対して通知されたもので，「ネットワーク通信」は Web ページの表示に対して通知されたものである．「ファイルのオープン」は Internet Explorer の実行ファイルを実行する際や，拡張機能の設定ファイルをインポートする際のオープンに対して通知されたものである．「プロセスの実行」は Internet Explorer のプロセスを実行して，Web ページを表示する機能に対して通知されたものである．

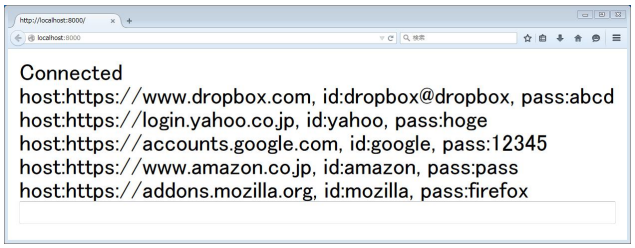

図 6 攻撃者側のブラウザ上に表示されたログイン情報

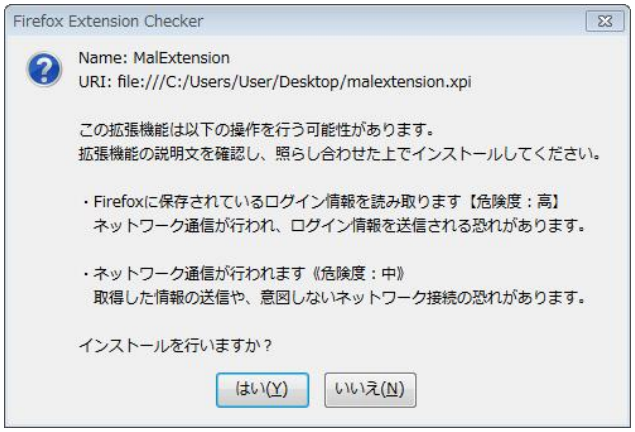

図 7 通知例 1

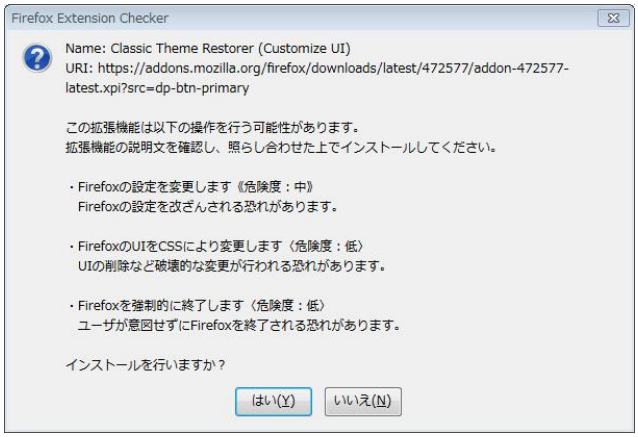

図 8 通知例 2

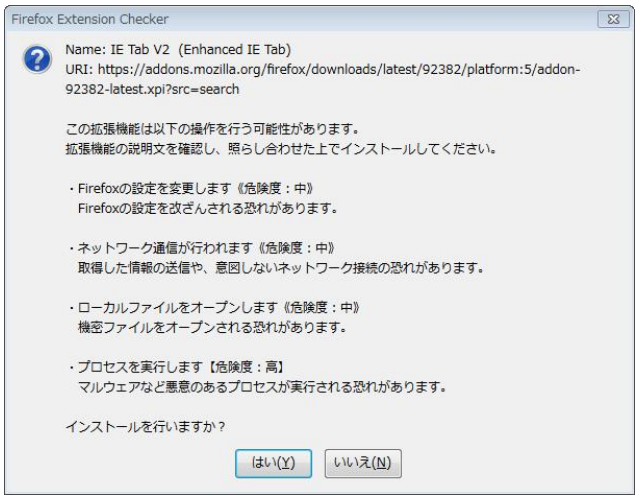

図 9 通知例 3

表 2　オーバーヘッドの計測結果

	JSFileSize		Overhead		
	Largest [KB]	Total [KB]	Download [ms]	Analyze [ms]	Total [ms]
Average	122	500	1589	1165	2754
LastPass	1151	1445	1696	15413	17110
Firebug	234	3930	1652	2303	3955

5.4　オーバーヘッド

通知システムの導入により発生するオーバーヘッドは，通知システムにより拡張機能のインストール処理が中断されてから，解析結果のダイアログが通知されるまでの時間である．そのオーバーヘッドを拡張機能本体のダウンロードに要する時間（Download）と，静的解析に要する時間（Analyze）に分割して計測した．計測対象は 2015 年 3 月 31 日時点でユーザ数トップ 50 の AMO で配布されている拡張機能である．それら 50 個の拡張機能に対する計測結果の平均値に加え，オーバーヘッドが最大となった LastPass Password Manager [13] に対する計測結果，静的解析の対象となるファイルサイズの合計（JSFileSize の Total）が最大となった Firebug [14] に対する計測結果を**表 2** に示す．オーバーヘッドの最大値は約 17 秒であり，一部の拡張機能に対しては 10 秒以上の時間を要したが，50 個の拡張機能を平均すると約 2.8 秒となった．オーバーヘッドの内，拡張機能本体のダウンロードに要する時間は約 1.6 秒で拡張機能本体のファイルサイズにかかわらず，どの拡張機能に対してもほぼ固定値であった．一方，静的解析に要する時間は，静的解析の対象となるファイル中で最大のファイルサイズ（Largest）によって大きく変動することが確認された．静的解析の対象となるファイルに，ファイルサイズの大きいものが含まれていない場合，Firebug のようにファイルサイズの合計が大きい場合でも，さほど静的解析に時間を要することはなかった．しかし，ファイルサイズの大きいものが含まれていた場合，文字列検索で大きなオーバーヘッドが生じ，LastPass のように静的解析である程度の時間を要することが確認された．なお，オーバーヘッドが 4 秒以内となった拡張機能は全体の 92%にあたり，ほとんどの拡張機能に対して数秒程度のオーバーヘッドで収まる結果となった．

6.　Firefox 拡張機能の署名

本章では，セキュリティ対策の一環として 2015 年に導入される予定である Firefox 拡張機能の署名について説明し，署名制度が通知システムに与える影響を考察する．

6.1　概要

Firefox 拡張機能は Mozilla の公式サイトである AMO で配布されているものがインストールされることが一般的である．AMO で配布されている拡張機能は，事前審査

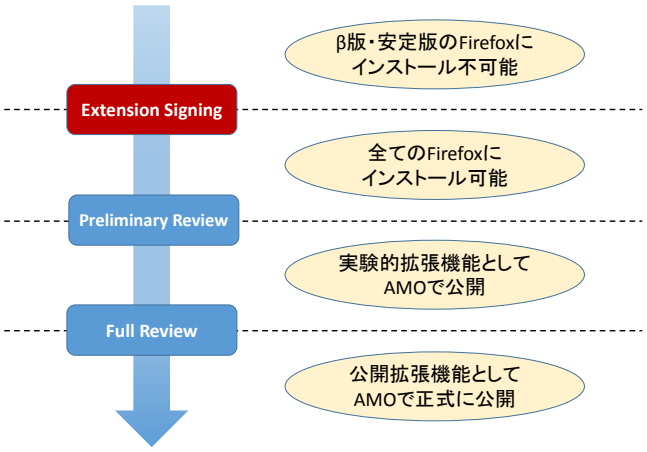

図 10　Mozilla による拡張機能の審査プロセス

（Preliminary Review）と本審査（Full Review）の 2 段階による審査が行われ，悪意を持つ拡張機能が AMO に登録されないように対策を講じている．一方，AMO で配布されている拡張機能以外の拡張機能は，第三者拡張機能や野良拡張機能と呼ばれ，制限なくサイドローディングによるインストールが可能である．もし悪意を持つ第三者拡張機能が発見された場合，1 章で述べたブロックリストに登録され，ブロックリストに登録された拡張機能はインストールを行うことが不可能となるという対策を Mozilla は講じている．ところが，悪意を持つ第三者拡張機能による攻撃は拡大しており，ブロックリストによる対策に加え，新たな対策の必要性が生じている．

そこで Mozilla は悪意を持つ拡張機能に対する新たなセキュリティ対策の一環として，Firefox 拡張機能に対し，署名（Extension Signing）の義務付けを決定した [15]．これは 2015 年 6 月にリリース予定の Firefox 39 から導入される予定である．署名制度が導入された後における，Mozilla による拡張機能の審査のプロセスを**図 10** に示す．この署名制度が導入されると，β 版の Firefox と一般的に使用されている安定版の Firefox では，署名付きでない拡張機能はインストールが不可能となる．拡張機能に署名を付与するには Mozilla による新たな審査を通過することが必要となる．署名付与のための審査の際に，悪意を持つ拡張機能を除外することで，セキュリティを向上させる手法である．

なお，Google により開発が行われている Google Chrome では，Google の公式サイトである Chrome ウェブストアで配布されている拡張機能のみインストール可能であり，第三者拡張機能のサイドローディングによるインストールを完全に制限している．しかし，Mozilla は拡張機能の開発において，オープン性や柔軟性を重視しており，サイドローディングを完全に制限するという手法は余計な制約であると考えている．セキュリティ面とオープン性や柔軟性のバランスを考慮した結果，Firefox 拡張機能では署名による手法を採用したと考えられる．

6.2 通知システムに与える影響

署名制度の目的は，悪意を持つ拡張機能を排除することであり，通知システムの目的の一つである悪意の操作の検出と役割が重なる部分が存在する．しかし，現状公開されている署名制度の内容を検証すると，依然として通知システムが必要であると考えられる．

ここで，Mozilla が署名制度の導入により防止すると明言している悪意の操作を列挙すると，「Web ページへの広告の挿入」，「悪質なスクリプトの埋め込み」，「ユーザの同意なしにホームページや検索エンジンの設定を変更すること」の 3 つが例示されている [15]．これら 3 つだけではなく，より多くの悪意の操作を署名付与のための審査によって防止すると思われるが，3 章で述べたように，Mozilla により提供される API 群による悪意の操作は非常に多岐に渡り，全ての悪意の操作を防止することは非常に困難であると考えられる．Mozilla も署名制度により全ての悪意の操作を防止できるわけではないと明言しており [15]，悪意を持つ拡張機能が署名付与のための審査を通過する可能性が十分に考えられる．そこで通知システムの一つの必要性として，審査を通過した悪意を持つ拡張機能が実行する操作を解析し，通知することが考えられる．

また，ある操作が悪意の操作に該当するか否かという認識や，実行を制限したい操作は個々人の状況によって差が生じる．そのため，もう一つの通知システムの必要性として，悪意の有無にかかわらず特定の操作や，特定の API の使用を解析し，通知することが考えられる．例えば，必要以上のネットワークの接続が望ましくない状況で Firefox 拡張機能を使用したい場合に，通知システムを使用することでインストールしようとしている拡張機能がネットワークの接続を必要とするのかを判断するという使途が考えられる．

7. まとめと今後の課題

本稿では，Firefox 拡張機能が実行しうる操作をユーザに通知するシステムを提案した．また，実装した通知システムを使用し，解析結果の通知例やオーバーヘッドを示した．そして，Firefox 拡張機能の署名について考察を行った．

今後の課題として，まず静的解析における誤検出の要因を解決するために，検出対象のメソッドやプロパティを呼び出したインスタンスの生成元が，検出対象の API であるかどうかを検証する仕組みを実装することが挙げられる．さらに，現状の悪質な挙動に繋がる操作の解析に加え，データフロー解析などと組み合わせることで，より具体的な悪意の操作を解析できるようにすることが挙げられる．また，Firefox 拡張機能の署名が公開され次第，どの程度の悪意の操作が署名付与のための審査を通過可能であるかを調査することや，悪意の操作の防止に関して通知システムと比較することが挙げられる．

参考文献

[1] Sampsa Rauti, Ville Leppanen. Browser Extension-Based Man-in-the-Browser Attacks against Ajax Applications with Countermeasures. *In Proceedings of the International Conference on Computer Systems and Technologies (CompSysTech)*, 2012.

[2] Blocked Add-ons :: Add-ons for Firefox
`https://addons.mozilla.org/en-US/firefox/blocked/`

[3] Wade Alcorn, Christian Frichot, Michele Orru. The Browser Hacker's Handbook, pp.315, 2014.

[4] Rezwana Karim, Mohan Dhawan, Vinod Ganapathy, Chung-chieh Shan. An Analysis of the Mozilla Jetpack Extension Framework. *In Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, 2012.

[5] Josh Marston, Komminist Weldemariam, Mohammad Zulkernine. On Evaluating and Securing Firefox for Android Browser Extensions. *In Proceedings of the MOBILESoft*, 2014.

[6] Ajin Abraham. Abusing, Exploiting and Pwning with Firefox Add-ons. *In Proceedings of the AppSec AsiaPac (APAC)*, 2013.

[7] Sruthi Bandhakavi, Samuel T. King, P. Madhusudan, Marianne Winslett. VEX: Vetting Browser Extensions for Security Vulnerabilities. *In Proceedings of the USENIX Security Symposium*, 2010.

[8] Hossain Shahriar, Komminist Weldemariam, Mohammad Zulkernine, Thibaud Lutellier. Effective detection of vulnerable and malicious browser extensions. *Computers & Security*, Vol. 47, pp. 66-84, 2014.

[9] Jiangang Wang, Xiaohong Li, Xuhui Liu, Xinshu Dong, Junjie Wang, Zhenkai Liang, and Zhiyong Feng. An Empirical Study of Dangerous Behaviors in Firefox Extensions. *In Proceedings of the Information Security Conference (ISC)*, 2012.

[10] Appendix C: Avoiding using eval in Add-ons - Mozilla — MDN
`https://developer.mozilla.org/en-US/Add-ons/Overlay_Extensions/XUL_School/Appendix_C:_Avoid_using_eval_in_Add-ons`

[11] Classic Theme Restorer (Customize UI) :: Add-ons for Firefox
`https://addons.mozilla.org/en-US/firefox/addon/classicthemerestorer/`

[12] IE Tab V2 (Enhanced IE Tab) :: Add-ons for Firefox
`https://addons.mozilla.org/en-US/firefox/addon/ie-tab-2-ff-36/`

[13] LastPass Password Manager :: Add-ons for Firefox
`https://addons.mozilla.org/en-US/firefox/addon/lastpass-password-manager/`

[14] Firebug :: Add-ons for Firefox
`https://addons.mozilla.org/en-US/firefox/addon/firebug/`

[15] Introducing Extension Signing: A Safer Add-on Experience — Mozilla Add-ons Blog
`https://blog.mozilla.org/addons/2015/02/10/extension-signing-safer-experience/`