# AndroidにおけるURLバーの切り替わり間隔に着目した利用者の意図しないWebサイトへの遷移の検知手法

折戸 凜太郎[1]　佐藤 将也[1]　山内 利宏[1]

**概要**：Android において，リダイレクトにより利用者を意図しない Web サイト（以降，悪性 Web サイト）へ誘導する攻撃が問題となっている．誘導先の Web サイトには，偽警告画面を表示する Web サイトやフィッシングサイトなどがある．対策として，URL のブラックリストを用いる手法があるものの，効率よく悪性 Web サイトへのアクセスを検知するのは難しい．また，我々が調査した限りでは，Android を対象として，リダイレクトにより悪性 Web サイトへ誘導する攻撃を検知する研究はない．そこで，Android において悪性 Web サイトへの遷移を検知するために，アクセシビリティサービスを用いた URL バーの切り替わり間隔に着目した検知手法を提案する．提案手法は，悪性 Web サイトの情報を事前に収集することなく，Android 上のブラウザアプリの URL バーの切り替わり間隔のみを用いて，高い精度で悪性 Web サイトへの遷移を検知できる．また，1 つアプリとして実現できるため，導入も容易である．本稿では，提案手法の設計と実現方式について述べ，また，提案手法の検知精度と性能オーバヘッドの評価結果を報告する．

# Detection Method of Transition to Unwanted Website Focusing on URL Bar Switching Interval in Android

Rintaro Orito[1]　Masaya Sato[1]　Yamauchi Toshihiro[1]

***Abstract:*** An attack that redirects the user to an unwanted website has become a problem in Android. These websites include the website displaying fake virus alert and phishing sites. As a countermeasure, although there is a countermeasure using URL black lists, it is difficult to detect the access to malicious website efficiently. In addition, as far as we have investigated, there is no method to detect the attack in Android. In this paper, we propose a method to detect transitions to unwanted websites focusing on the switching interval of the URL bar by AccessibilityService. The proposed method can detect with high accuracy without collecting information in advance, and can be easily introduced as an app. In this paper, we describe its design and implementation, and the evaluation results of the detection accuracy and performance overhead.

## 1. はじめに

モバイル端末の普及により，モバイル版 Web ブラウザの使用率は PC 版 Web ブラウザの使用率を上回っており，これに伴いモバイル端末を対象としたマルウェアが増加している．PC を対象としたマルウェアのサンプルは，200 万件へ到達までに 20 年間を要した一方で，モバイル端末を対象としたマルウェアのサンプルは同じ数に到達するまでに 5 年間しか要さなかった [1].

Android 端末を対象とした攻撃に，利用者を意図しない Web サイト（以降，悪性 Web サイト）へ誘導する攻撃が存在する．この攻撃では，利用者が遷移元サイトに Web アクセスした際に，自動的もしくは画面のタップなどの操作を契機として，複数の Web サイト（以降，経由サイト）をリダイレクトによって経由することで，目的の悪性 Web サイトへ誘導する．複数の Web サイトを経由する点は，Drive-by Download 攻撃と類似している [2].

しかし，モバイル端末は PC とは異なり，Web サイト上で利用者の許可を得ることなく，ソフトウェアを自動的にインストールさせることはできない．このため，モバイル

[1] 岡山大学 大学院自然科学研究科
Graduate School of Natural Science and Technology, Okayama University

端末における攻撃は，誘導先の Web サイトで利用者を欺くことで，広告収入の獲得や個人情報の奪取を目的としている [3]．たとえば，誘導先の Web サイトには，偽警告画面を表示する Web サイトやフィッシングサイトなどがあり，これらの Web サイトでは偽の警告や偽の懸賞当選を表示する手段を用いている [4], [5], [6]．

このように利用者を欺くことで，利用者が自らマルウェアに感染してしまう行動や個人情報を流出してしまう行動を促す手口は，Android 端末を対象としている主要な攻撃方法の 1 つである [7]．文献 [8] では，インターネット利用中に偽の警告画面へ誘導されてしまう攻撃に関する相談件数が，2018 年 5 月から急激に増加していることを注意喚起している．また，この攻撃では，企業のロゴを無断使用することで，本物の警告画面を装う場合がある．

文献 [9] では，悪性 Web サイトへ Web アクセスしてしまった場合であっても，利用者が危機的な状況におかれていることを認識できない可能性が高いと警告している．したがって，スマートフォンの利用者をこのような攻撃から保護するためのセキュリティ機構が必要である．

対策として，URL のブラックリストを用いる手法がある．しかし，膨大な数の悪性 Web サイトが存在すること，および悪性 Web サイトの URL とドメイン名は短期的に変更されることから，この手法で，多くの悪性 Web サイトへの Web アクセスを防止することは簡単ではない [4], [10], [11]．Android におけるセキュリティアプリの多くは，ブラックリスト方式を採用しており，検知率が高くないという報告がある [12]．

この対策以外に，リダイレクトによって複数の Web サイトを経由する特徴に着目した先行研究がある [2], [13]．しかし，これらの研究は Android を対象としていない．また，我々が調査した限り，上記の先行研究を含め，Android を対象とした悪性 Web サイトへ誘導する攻撃の対策を提案している研究は，見つかっていない．

Windows などの PC 向けの OS では，ウイルス対策ソフトウェアが Web ブラウザにおける通信や利用者の操作を監視して，危険な Web アクセスや操作を事前に検知することができる．一方，Android では，セキュリティアプリが，Web ブラウザアプリの通信を監視したり，API 呼出しを制御したりできないため，実施できるセキュリティ対策が限られる．したがって，Android を対象としていない既存の対策を，そのまま Android に適用することは難しい．

そこで，本研究では，Android を対象として，リダイレクトによって利用者を意図しない Web サイトへ誘導する攻撃を，アクセシビリティサービスを用いて検知する手法を提案する．我々は，過去の Android における偽警告画面を表示する Web サイトへの Web アクセスの分析結果 [14] や，2 章で述べる悪性 Web サイトへの誘導の調査結果により，意図しない Web サイトに誘導する際には，最低でも 2 回のリダイレクトが起こることを明らかにした．提案手法は，この誘導時の複数回のリダイレクト発生間隔に着目して，悪性 Web サイトへの遷移を検知する手法である．提案手法は，ブラックリストなどを用いることなく，悪性 Web サイトへの遷移を高い精度で検知できる特徴がある．具体的には，提案手法は，アクセシビリティサービスを用いて，Google Chrome の URL バーの切り替わる時間の間隔を観測することで，悪性 Web サイトへの遷移を伴うリダイレクトによる Web ページの切り替えを判断する．連続的に発生するリダイレクトによる Web ページの切り替えは，利用者の通常の利用による切り替えよりも明らかに間隔が短いことを利用して，利用者を悪性 Web サイトへ誘導する攻撃を検知する．

本研究の主な貢献は，以下の通りである．

(1) 本研究は，利用者を意図しない Web サイトへ誘導する攻撃に関して，Web ページの遷移間隔に基づき検知する新たな手法を提案した．この手法は，事前に悪性 Web サイトの情報を収集することなどを必要とせず，Web ブラウザの URL バーの内容の変化間隔に着目したシンプルで高精度な検知手法である．
(2) リダイレクトを発生させるコードの種類によらず，提案手法は，URL バーの内容の変化を検知し，悪性 Web サイトへの誘導を検知できる．つまり，Web アクセス時に HTML ファイルや JavaScript コードなどを解析する必要がない利点がある．
(3) 提案手法は 1 つの Android アプリとして実現しているため，利用者は通常のアプリと同じインストール手順で導入でき，導入は簡単である．
(4) 提案手法の検知精度を，他のセキュリティアプリと比較した結果から，提案手法は利用者を意図しない Web サイトへ誘導する攻撃への対策として，有用であることを明らかにした．

## 2. Android における利用者の意図しない Web サイトへの誘導の調査

WebView の観測機構 [15], [16] を用いて，悪性 Web サイトに誘導される際の Web アクセスを観測し，悪性 Web サイトへのリダイレクトの特徴を調査した．なお，調査は，Web 検索，および Twitter や Facebook の投稿から発見した 10 件の遷移元サイトを対象に実施した．この調査と文献 [14] の結果から明らかにした Android における遷移元サイトから悪性 Web サイトへの遷移の流れを**図 1** に示す．以下に，発見した 6 つの特徴について述べる．

**（特徴 1）** 2 つ以上の Web サイトを経由して誘導

図 1 に示すように，今回調査したすべての遷移元サイトにおいて発生する 2 回以上のリダイレクトによって悪性 Web サイトへ誘導されることを明らかにした．

**（特徴 2）** JavaScript コードなどにより，経由サイトの

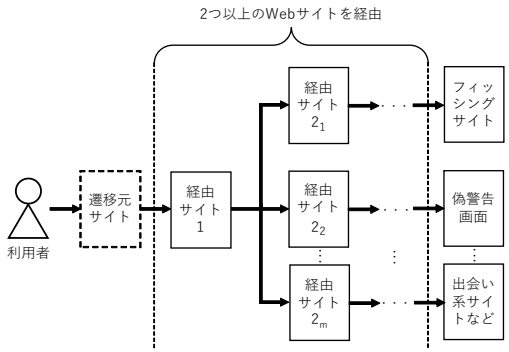


**図 1** 利用者の意図しない Web サイトへの遷移の流れ

ダウンロードと表示の完了前にリダイレクトが発生

経由サイト 1 以降のリダイレクトは，Web ページの読み込みと表示が終わる前に発生し，次の経由サイトもしくは悪性 Web サイトにアクセスするため，通常よりも短時間の間に，多くのリダイレクトが起こる．

**（特徴 3）** リダイレクト時に異なる URL を生成

1 つ目の経由サイトでは，Web アクセスするごとに異なる URL を生成し，生成した URL をリダイレクト先とする．このため，攻撃が実行されるごとに URL が異なる次の経由サイトへ遷移する．これにより，2 つ目の経由サイトの URL や誘導先の悪性 Web サイトの種類は，攻撃が実行されるごとに変化する．

**（特徴 4）** 閲覧履歴を残さないリダイレクト方法

2 つ目以降の経由サイトへリダイレクトする際は，JavaScript の window.location.replace() メソッドを用いて，閲覧履歴を残すことなく遷移することが多い．閲覧履歴が残らないため，Web ブラウザの「戻る」ボタンにより前のページに戻ることができなくなる．

**（特徴 5）** Cookie の有無によるリダイレクト発生の違い

遷移元サイトで実行される JavaScript コードの分析によって，遷移元サイトでは常にリダイレクトが発生するとは限らず，遷移元サイトの Cookie を保持していない場合などの特定の条件を満たしているときだけにリダイレクトが発生することを確認した．

**（特徴 6）** 複数のリダイレクトコード

リダイレクトは HTML，JavaScript，および PHP のコードによって実行されることを確認した．また，異なるコードでリダイレクトは実行されても，画面上の Web ページの遷移の振る舞いに違いがないことも確認した．

## 3. 提案手法

### 3.1 目的と考え方

本研究の目的は，悪性 Web サイトの URL などを事前に収集することなく，Android において利用者の意図しない Web サイトへ誘導する攻撃を検知し，警告を表示する手法の実現である．これは，2 章で述べた（**特徴 3**）のように，経由サイトの URL はアクセス毎に変化するため，Android における多くのセキュリティアプリが採用するブラックリスト方式だけでは，効率の良い検知が難しいためである．

また，（**特徴 5**）と（**特徴 6**）で述べたように，悪性 Web サイトへの遷移は，JavaScript コードによる Cookie の制御や，HTML，JavaScript，および PHP のリダイレクトコードを用いて実行されている．このため，それぞれのコードを実行時に解析し，対策することも難しい．特にこの攻撃に用いられる多くの JavaScript コードは，難読化が施されており，リダイレクトを生じさせる特定の JavaScript コードを識別することは難しい．

以上のことから，事前に悪性 Web サイトの情報の収集，および実行時のコード解析を行うことなく，悪性 Web サイトに誘導するリダイレクトを検知する手法を検討する．

このために，（**特徴 1**）と（**特徴 2**）に示したリダイレクトによって 2 つ以上の Web サイトを短時間に経由して悪性 Web サイトへ誘導する特徴を検知する手法を提案する．

以降では，Google Chrome を提案手法の監視対象とした場合を例として，説明する．

### 3.2 課題

**設計方針:** 提案手法の機能を追加するプログラムを検討する必要がある．一つの方法として，Android のベースとなる Linux カーネルや Android フレームワークに通信の監視，もしくは Web アクセスを監視する機能を追加する方法がある．しかし，これらを改変して，一般の利用者が端末に導入するのは，容易ではない．

そこで，提案手法の機能を 1 つのアプリとして実現し，アプリとしてインストールして導入できる方法を採用する．

上記の設計方針における設計の課題を以下に示す．

**（課題 1）** Android において他のアプリの Web アクセスを把握する方法

Android では，他のアプリの通信を監視できない．このため，提案機能を有するアプリが，監視対象のアプリの Web アクセス先を把握する方法を検討する必要がある．

**（課題 2）** リダイレクトによる複数の Web サイトの経由を検知する方法の検討

監視対象ブラウザにおいてリダイレクト発生時に，Web サイトへのアクセスが発生したことを，提案機能のアプリ内で把握する方法の検討が必要である．また，その情報で悪性 Web サイトへの遷移を検知する手法の検討が必要である．

**（課題 3）** 悪性 Web サイトへの遷移を検知したことを利用者に通知する手段の検討

悪性 Web サイトへの遷移を検知した際，検知内容を利用者に通知する必要がある．利用者へ通知する手段によっては，利用者が危機的状況におかれていることを意識できない可能性がある [17]．このため，利用者に危険であることを意識させる通知手段を検討する必要がある．

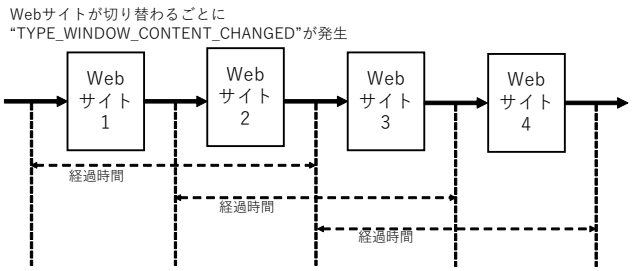


**図 2** 検知に用いる URL バーの切り替わり時間の間隔の算出方法

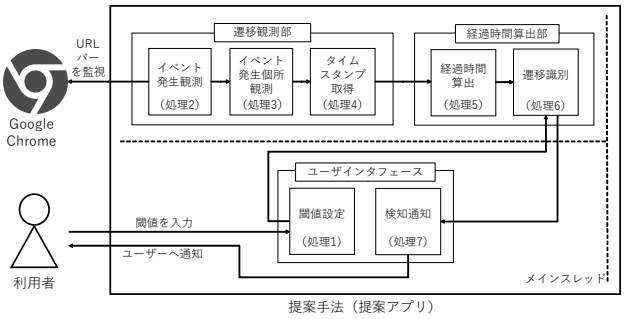


**図 3** 提案手法の全体像

### 3.3 課題への対処

#### 3.3.1 Android において他のアプリの Web アクセスを把握する手法

提案手法は，リダイレクトによる複数の Web サイトの経由を検知するために，アクセシビリティサービスを用いて URL バーの切り替わりを観測する．アクセシビリティサービスは，画面上の状態を観測し，その観測した状態に対して様々なサービスを提供する．また，アクセシビリティサービスの機能の 1 つにイベントと呼ばれる利用者の端末操作や画面上の動きを観測する機能がある．

#### 3.3.2 リダイレクトによる複数の Web サイトの経由を検知する方法の検討

提案手法は，アクセシビリティサービスのイベントを検知する機能により，リダイレクトによる複数の Web サイトの経由を識別する．Google Chrome では，Web サイトが切り替わるごとに，URL バーに表示されている URL の内容が変化する．URL バーの内容が変化するたびに，アクセシビリティサービスによって検知できるイベント（以降，アクセシビリティイベント）である “TYPE_WINDOW_CONTENT_CHANGED” が発生する．このアクセシビリティイベントは，View 上に表示されているテキストの内容が変更したときに発生する．

悪性 Web サイトへは，リダイレクトによって複数の Web サイトを経由した後に遷移するため，悪性 Web サイトへ遷移する際は，このイベントが短時間に連続して発生する．また，リダイレクトは，Web サイトの HTML ファイル全体の読み込みが完了する前に実行されるため，悪性 Web サイトへの遷移する際は，リダイレクトが短時間のうちに連続して発生する．そこで，短時間で URL バーが連続して切り替わることを検知できた場合，利用者の意図しない Web サイトへの遷移を検知できたといえる．

悪性 Web サイトへ遷移する際，2 つ以上の Web サイトを経由する（図 1）．このため，遷移元サイトへのアクセス直後から，悪性 Web サイトへの到達までに，3 回以上の “TYPE_WINDOW_CONTENT_CHANGED” が発生する．

以上より，提案手法では，**図 2** に示すように，URL バーの切り替わりが発生するごとに，2 回前の切り替わり発生時刻からの経過時間（以降，経過時間）を算出する．2 回前のリダイレクトからの経過時間が閾値よりも短い場合，悪性 Web サイトへ遷移していると判断する．なお，設定する閾値については 4.2 節に後述する．

#### 3.3.3 悪性 Web サイトへの遷移を検知したことを利用者に通知する手段の検討

利用者が被害に合うことを防止するには，悪性 Web サイトを画面上に表示する前に，警告を表示する必要がある．このため，提案手法は，悪性 Web サイトへの遷移を検知した際，フォアグラウンドに表示しているアプリを，Google Chrome から提案手法が実現されているアプリ（以降，提案アプリ）に切り替える．切り替えた後に，提案アプリ上で検知したことを利用者に通知する警告文を表示する．これにより，悪性 Web サイトの表示を見る前に，警告文を読むことができ，自身が危険な Web サイトへ誘導されたことを認識できる．

なお，Android では，他のアプリの通信を監視して，アクセスをブロックすることができない．このため，この対処では，Web ブラウザは悪性 Web サイトをバックグラウンドで読み込んでいる可能性があるものの，フォアグラウンドでは，提案アプリの警告を先に利用者に表示することで，悪性 Web サイトを利用者が利用することを防ぐ．

### 3.4 全体像

**図 3** に，提案手法の全体像を示し，説明する．提案手法では，アクセシビリティサービスを用いて URL バーを監視する機能を遷移観測部とし，経過時間を算出することで悪性 Web サイトへの遷移を検知する機能を経過時間算出部とする．また，検知した際に提案アプリに自動的に切り替え，提案アプリ上で警告文を表示する．

Android アプリの UI に関する処理は，メインスレッドと呼ばれるスレッド上で実行される．提案手法は，上記の処理のうち，警告文の表示をメインスレッド上で行う．また，提案アプリのメイン画面には，閾値を利用者が自由に設定できるフォームを設けている（4.2 節に後述）．

### 3.5 提案手法の処理流れ

**図 4** に提案手法の処理のフローチャートを示す．

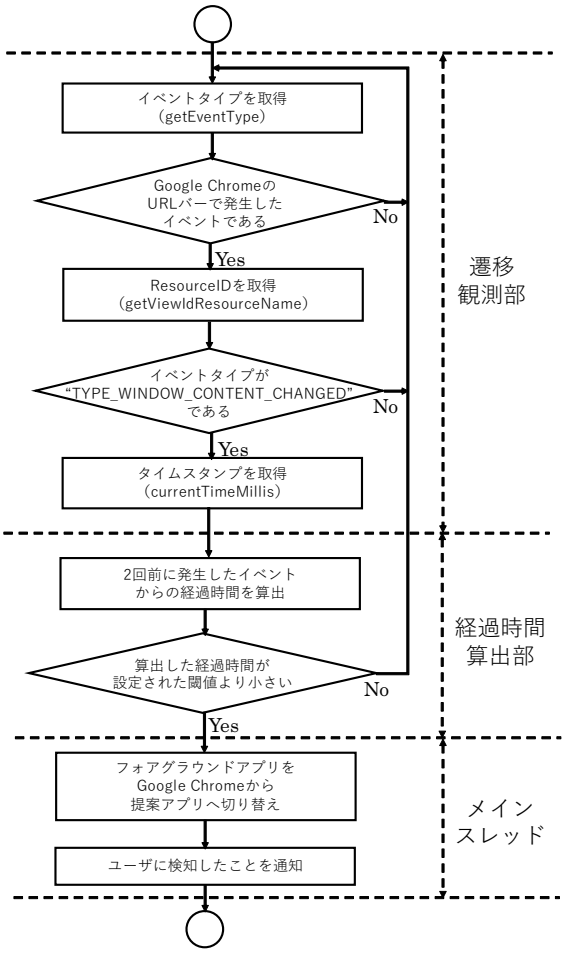


**図 4** 提案手法の処理のフローチャート

**（処理 1）** 利用者は提案アプリのメイン画面において，経過時間の閾値を設定する．

**（処理 2）** 観測したアクセシビリティイベントが “TYPE_WINDOW_CONTENT_CHANGED” であるか否か確認する．

**（処理 3）** “TYPE_WINDOW_CONTENT_CHANGED” の発生個所が Google Chrome の URL バー上であるか否か確認する．

**（処理 4）** “TYPE_WINDOW_CONTENT_CHANGED” が発生した際のタイムスタンプを取得する．

**（処理 5）** **（処理 4）**で取得したタイムスタンプを用いて経過時間を算出する．

**（処理 6）** 算出した経過時間が，定めた閾値より小さいか否かを判定する．

**（処理 7）** 経過時間が閾値より小さい場合，フォアグラウンドアプリを提案アプリに切り替え，利用者に悪性 Web サイトへの遷移を検知したことを通知する．

## 4. 実現方式

### 4.1 提案手法の実現方式

onAccessibilityEvent() メソッドを利用して，提案手法を実現する．このクラスは，アクセシビリティサービスを用いて検知できるすべてのイベント発生時に呼び出される．また，onAccessibilityEvent() メソッド内では，プロパティと呼ばれる検知したイベントに関する情報を取得できる．

まず，提案手法は，getEventType() メソッドにより，イベント名を取得する．発生したアクセシビリティイベントが “TYPE_WINDOW_CONTENT_CHANGED” の場合，getViewIdResourceName() メソッドにより，イベントが発生した View の Resource ID を取得し，検知した “TYPE_WINDOW_CONTENT_CHANGED” が Google Chrome の URL バーで発生したか否かを判定する．Resource ID は，View が含まれているアプリのパッケージ名，および View の ID 名によって構成される．

Google Chrome の URL バー上でイベントが発生したと判定した場合，System.currentTimeMillis() メソッドにより，イベント発生時刻を取得する．次に，保存しておいた 2 回前のイベント発生時刻と，取得した時刻を用いて，経過時間を算出する．経過時間が設定した閾値よりも小さい場合，提案アプリをフォアグラウンドへ切り替える．この後，提案アプリ上で利用者の意図しない Web サイトへの遷移を検知したことを通知する．

提案手法は，Google Chrome を対象として提案アプリを作成した．他のブラウザについては，そのブラウザの URL バーの Resource ID を用いることで，同様に URL バーの切り替わりを検知し，意図しない Web サイトへの遷移を検知できると考えている．

**表 1** 評価環境

| | |
|---|---|
| OS | Android 8.0 (Oreo) |
| CPU | Kryo 385, 2.8 GHz (8 core) |
| メモリ | 4 GB |
| 通信規格 | AXGP (Softbank 4G) |
| Google Chrome | 75.0.3770.143 |

### 4.2 閾値の検討

#### 4.2.1 閾値の設定における課題

実現における課題として，悪性 Web サイトへ遷移を検知する経過時間の閾値の検討がある．閾値が大きすぎる場合，通常の Web ページの切り替えを誤検知してしまう可能性がある．一方，閾値が小さすぎる場合，悪性 Web サイトへの遷移を検知できない可能性がある．

ここでは，適切な閾値を定めるために，悪性 Web サイトへ遷移する際に要する時間を測定した．また，通常の Web ページについて，利用者ができるだけ速くタップして切り替えた場合に要する切り替わり時間を測定した．この 2 つの結果の比較から，誤検知を低減する閾値の設定方法について述べる．評価環境を**表 1** に示す．

#### 4.2.2 実験方法

**（実験 1）** 悪性 Web サイトへ遷移する際の経過時間の測定

表 2　実験用サイトの HTTP レスポンスの合計サイズ（単位：KB）

| 実験用サイト 1 | 実験用サイト 2 | 実験用サイト 3 |
|---|---|---|
| 2,069 | 1,388 | 2 |

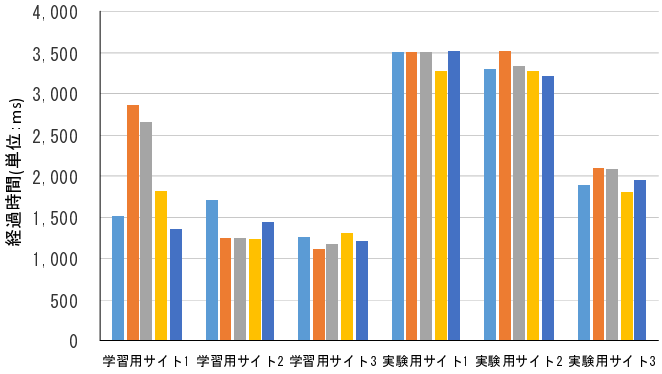


**図 5**　経過時間の算出結果

(1) 遷移元サイトにおいて，悪性 Web サイトへの遷移を発生させる．
(2) 悪性 Web サイトへの遷移までの経過時間を算出した．

この実験を 3 つの遷移元サイト（以降，学習用サイト）について，5 回ずつ実施した．各学習用サイトで 5 回ずつ実施した理由は，遷移が発生するごとに経由サイト，および誘導先の悪性 Web サイトが異なるためである．

**（実験 2）** 通常の Web ページの切り替えによる Web サイトが遷移する際の経過時間の測定

(1) ある Web サイトにおいて，リンクをタップすることで Web ページの切り替えを行う．
(2) (1) の操作を連続して行う．
(3) (1) および (2) の操作を可能な限り速く行い，Web ページが 3 回切り替わった際の経過時間を算出した．

この実験を 3 つの Web サイト（以降，実験用サイト）について，5 回ずつ実施した．

また，**（実験 2）**に用いた Web サイトの HTTP レスポンスの合計サイズを**表 2** に示す．HTML のメインリソースおよび，サブリソースを含むレスポンスの合計サイズにより，Web サイトの表示に要する時間は変化することを考慮し，**（実験 2）**では表 2 に示す HTTP レスポンスの合計サイズが異なる 3 つの Web サイトを用いた．実験用サイト 3 には，レスポンスサイズが小さいサイトを用いた．これは，サイズが小さいサイトへの切り替え時間が短いので，タップによりどれだけ短い間隔で Web サイトを切り替えられるのかを評価するためである．なお，レスポンスサイズの測定には，Google が提供するツール [18] を利用した．

#### 4.2.3　実験結果

測定した経過時間を Web サイトごとに試行した順に並べたものを**図 5** に示す．図 5 から，以下のことが分かる．

**（実験 1）**で算出した経過時間の多くは，**（実験 2）**で算出した経過時間よりも短い．これより，多くの場合，悪性 Web サイトへ遷移する際の URL バーの切り替わり時間間隔は，リンクをタップした際の URL バーの切り替え時間間隔よりも短い．しかし，レスポンスサイズが小さい実験用サイト 3 では，悪性 Web サイトへの遷移との切り替え時間間隔の差は小さい．利用者が意図的にレスポンスサイズの小さい Web サイトで連続してタップして，短い間隔で Web サイトを切り替えることは少ないため，このような場合はあまり起こらないと推察できる．

一方で，戻るボタンを連続してタップした場合には，短い間隔で Web サイトの遷移が連続して発生する．この対策として，Web サイトの遷移が戻るボタンのタップで起きたか否かを判断することで対処できる．戻るボタンのタップで起きた判断した場合には，悪性 Web サイトへの遷移として検知しないことで，誤検知を防ぐことができる．

なお，Google Chrome については，アプリ内で戻るボタンが実現されておらず，Android 端末の戻るボタンが利用される．Android 端末の戻るボタンの押下は，onBackPressed() メソッドにより検知できる．一方，Google Chrome 以外のブラウザアプリには，戻るボタンがアプリ内に表示されているものがある．この場合は，アクセシビリティサービスによって，戻るボタンの押下を検知できる．

また，学習サイト 1 において算出した 2 つの経過時間は，2,500ms を超えている．これは，攻撃者が JavaScript の setTimeout() メソッドにより，悪性 Web サイトへ遷移する際のリダイレクトの発生を 1,000ms 遅らせる処理を行っているためである．

以上の結果から，評価環境においては，悪性 Web サイトへの遷移を制御された場合に対応でき，見逃しが生じない 3,000ms が閾値として適切であると推察する．以降は，この閾値を評価に用いた．なお，適切な閾値の値は，スマートフォン本体の性能や通信環境などにより，変化すると考えられる．このため，提案アプリにおいては，利用者が利用環境に合わせて閾値を調整できる機能が必要である．

### 4.3　導入方法

提案手法は，1 つの Android アプリに実現している．このため，提案手法は通常の apk ファイルと同様に多くの Android 利用者が容易に導入できる．また，アクセシビリティサービスは，Android バージョンによって検知できるイベントや取得できる情報が異なる．提案手法が利用するアクセシビリティサービスの機能は，対象の Android 端末のバージョンが Android 4.3 以上で動作し，多くの端末に対応できる．

## 5.　評価

### 5.1　評価内容と評価環境

提案手法の有用性，および導入によって生じるオーバヘッドを検証するために，以下の評価を行った．評価環境は，表 1 と同様である．

**表 3** 提案手法および各セキュリティツールの検知精度

| | 提案手法 | GSB | アプリ 1 | アプリ 2 |
|---|---|---|---|---|
| 検知件数 | 25 件 | 0 件 | 1 件 | 0 件 |

**表 4** 各処理に要する時間の平均値（単位：ms）

| 処理 2 | 処理 3 | 処理 4 | 処理 5 | 処理 6 | 処理 7 |
|---|---|---|---|---|---|
| 0.0028 | 0.0013 | 0.0024 | 0.00069 | 0.00072 | 4.3 |

**（評価 1）**提案手法の検知精度

Google Safe Browsing（以降，GSB），および Play ストアにおいて利用者評価の高い 2 つのセキュリティアプリ（以降，アプリ 1，2）を用いて，提案手法と既存のセキュリティアプリの悪性 Web サイトへの遷移の検知精度を評価した．アプリ 1 は，商用の有料アプリであり，アプリ 2 は無料のアプリであり，いずれのアプリもインストール数は，5,000,000 件を超えている．

評価には，新しい悪性 Web サイトを用いて評価するため，Twitter の Streaming API を用いるモバイル向けの悪性 Web サイト探索手法 [19] により，2019 年 7 月 7 日に収集した 5 つの遷移元サイトを利用し，8 月 7 日に実験した．

遷移元サイトでは，悪性 Web サイトへのリダイレクトが発生するごとに経由サイトや悪性 Web サイトの URL が変化するため，各遷移元サイトにおいて悪性 Web サイトへのリダイレクトを 5 回発生させた．評価では，提案手法，および各セキュリティツールが悪性 Web サイトへの遷移を検知できたか否かを確認した．なお，GSB，アプリ 1，およびアプリ 2 は，経由サイトまた悪性 Web サイトのいずれかを検知した場合，悪性 Web サイトへの遷移を検知したと判断した．

**（評価 2）**提案手法に要する処理時間の測定

遷移元サイトに Web アクセスした後から，悪性 Web サイトへの遷移を検知するまでの各処理（図 3）に要する時間を測定した．なお，（処理 1）は利用者が任意のタイミングで実行する処理であり，悪性 Web サイトへの遷移を検知する処理ではないため，今回の評価では測定していない．

### 5.2 提案手法の検知精度

提案手法，GSB, アプリ 1，およびアプリ 2 の検知数（最大 25 件）を**表 3** に示す．表 3 から，提案手法はすべての悪性 Web サイトへの遷移を検知できていることがわかる．一方，GSB，アプリ 1，及びアプリ 2 については，アプリ 1 が 1 件検知したのみで，他の 2 つは検知できなかった．

GSB，アプリ 1，およびアプリ 2 の偽警告画面の検知精度については，文献 [12] でも報告している．このときの評価では，20 件の偽警告画面に対し，それぞれ 0 件，13 件，0 件の検知数であった．本稿の結果と比較すると，アプリ 1 の検知率の低下が大きい．これは，今回は Twitter でツイートされた新しい悪性 Web サイトを評価に用いたためだと推察している．一方で，提案手法はすべて検知できたことから，事前に悪性 Web サイトを収集することなく，検知できる提案手法の有用性が高いことが分かる．

### 5.3 提案手法に要する処理時間の測定

測定した各処理に要する時間の平均を**表 4** に示す．表 4 に示す処理のうち（処理 2）はアクセシビリティサービスによって検知できるすべてのイベントが発生するごとに実行される．今回，オーバヘッドを測定した 5 回の実験では，平均 56 回実行されており，これは他の処理と比べて実行される回数が多い．しかし，（処理 2）を 1 回実行するために要する時間は，表 4 に示すように極めて短いことから実行回数が多いことは問題とはならない．

また，(処理 3) は平均 24 回実行されているものの，これも（処理 2）と同様に 1 回実行するために要する時間は極めて短いことから実行回数が多いことは問題とはならない．（処理 4），(処理 5) および (処理 6) は，Google Chrome の URL バー上で “TYPE_WINDOW_CONTENT_CHANGED” が発生した場合のみ実行される処理であり，実行回数は少ない．また，表 4 より 1 回実行するために要する時間は短いことが分かる．（処理 7）は，アプリを切り替える処理（UI の変更）であるため，処理時間が他の処理と比べて長くなっているものの，4.3ms と処理時間が短いため問題ない．

以上のことから，提案手法に要する処理時間は短く，利用者の処理に問題となるような影響がないことが分かる．

## 6. 関連研究

利用者の意図しない Web サイトへ遷移する際に，リダイレクトによって複数の Web サイトを経由する特徴に着目した悪性 Web サイトの検知手法がいくつか存在する．

文献 [2] では，Drive-by Download 攻撃で発生する多段のリダイレクト情報を高対話型ハニークライアントによって自動的に収集し，リダイレクト情報の構造的類似性に基づいて悪性 Web サイトを検知する手法を提案している．この手法では，収集したリダイレクト情報から特徴量を算出し，悪性 Web サイトを検知するための教師あり機械学習に適用する．

文献 [13] では，悪性 Web サイトへ遷移する際の多段のリダイレクトの流れを表すリダイレクトグラフを大量に作成し，このグラフの特徴に基づいた検知手法を提案している．このグラフの作成に，経由サイトおよび誘導先の Web サイトの referrer を必要とする．referrer は，HTTP リクエストに含まれる情報であり，その Web サイトへ Web アクセスする際のリンク元の Web サイトの URL を示す．

これらの文献は，リダイレクトによって複数の Web サイトを経由する特徴に着目しているものの，Android に着目していない．Android では，Web ブラウザの外部から通信を監視することができず，実施できる対策が限られている．このため，リダイレクト情報の特徴量を用いた検知手

法は適用できない．

また，悪性 Web サイトや経由サイトは，HTML ファイル内に “noreferrer” を記述することで，referrer の送信を制御している場合がある．このため，リダイレクトグラフを用いた手法では，悪性 Web サイトへの遷移をすべて検知することはできない．

一方，提案手法は，Android の機能であるアクセシビリティサービスを用いて，通信ではなく画面上の動作を監視することで利用者の意図しない Web サイトへの対策を実現している．また，URL バーの切り替わりのみを観測することで悪性 Web サイトへの遷移を検知できるため，referrer の制御や JavaScript の難読化といった攻撃者が意図して行う情報の操作の影響を受けない利点がある．

## 7. おわりに

Android において，利用者を意図しない Web サイトへ誘導する攻撃への対策として，アクセシビリティサービスを用いた URL バーの切り替わり間隔に着目した検知手法を提案した．提案手法の Google Chrome の URL バーの切り替わる時間の間隔を算出し，この算出した値をもとに悪性 Web サイトへの遷移が発生しているか否かを識別する．また，提案手法は，Android アプリケーションとして実現するため，すべての利用者にとって導入が容易である．

提案手法の検知精度を GSB，および 2 つのセキュリティアプリと比較した結果から，提案手法は他のものよりも高精度で検知でき（25 件すべて検知），悪性 Web サイトへ誘導する攻撃への対策として有用であることを示した．また，性能オーバヘッドも小さいことも示した．

残された課題として，Google Chrome 以外の Web ブラウザアプリへの適用がある．

**謝辞** 本研究成果は，国立研究開発法人情報通信研究機構（NICT）の委託研究「Web 媒介型攻撃対策技術の実用化に向けた研究開発」により得られたものです．

### 参考文献

[1] Mobile Threat Report: McAfee Mobile Threat Report Q1, available from ⟨https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobilethreat-report-2018.pdf⟩ (accessed 2019–08–08).

[2] 芝原俊樹，八木毅，秋山満昭，高田雄太，矢田健：リダイレクトの構造的類似性に基づく悪性 Web ページ検知手法，コンピュータセキュリティシンポジウム 2015（CSS2015）論文集，Vol.2015，No.3，pp.496–503 (2015).

[3] Meridith Levinson: Mobile Malware: Beware Drive-by Downloads on Your Smartphone, CIO FROM IDG, available from ⟨https://www.cio.com/article/2397969/mobile-malware--beware-drive-by-downloads-on-your-smartphone.html⟩ (accessed 2019–08–08).

[4] R. Aravindhan et al.: Certain investigation on web application security: Phishing detection and phishing target discovery, Proceedings of the 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), (2016).

[5] DigitalArts：正規の Web サイトを改ざんし偽警告や偽当選サイトへ誘導する攻撃を確認，入手先 ⟨https://www.daj.jp/security_reports/190613_1/⟩ (参照 2019–08–08).

[6] 利穂虹希，折戸凜太郎，佐藤将也，山内利宏：Android を対象とした利用者の意図しない Web サイトの分類，コンピュータセキュリティシンポジウム 2019（CSS2019）.

[7] Wandera: Android Malware: 4 Ways Hackers are Infecting Phones with Viruses, available from ⟨https://www.wandera.com/malware-on-android/⟩ (accessed 2019–08–08).

[8] 独立行政法人情報処理推進機構 セキュリティセンター：偽のセキュリティ警告によって有償の「ソフトウエア購入」や「サポート契約」をしてしまう相談が増加中〜インターネット利用中に表示される偽の警告画面にだまされないで！〜，情報セキュリティ安心相談窓口だより，入手先 ⟨https://www.ipa.go.jp/security/anshin/mgdayori20180718.html⟩ (参照 2019–08–08).

[9] 日景奈津子，カールハウザー，村山優子：情報セキュリティ技術に対する安心感の構造に関する統計的検討，情報処理学会論文誌，Vol.48, No.9, pp.3193–3203 (2007).

[10] 尼子雄大，高田哲司：情報視覚化による Drive-by Download 攻撃対策の一検討，情報処理学会研究報告，Vol.2014-CSEC-64，No.41，pp.1–7 (2014).

[11] 笠間貴弘，井上大介，衛藤将史，中里純二，中尾康二：ドライブ・バイ・ダウンロード攻撃対策フレームワークの提案，コンピュータセキュリティシンポジウム 2011（CSS2011）論文集，Vol.2011，No.3，pp.780–785 (2011).

[12] 折戸凜太郎，佐藤将也，山内利宏：Android 向けセキュリティアプリにおける悪性 Web サイト検知率の調査，第 18 回情報科学技術フォーラム（FIT2019），L-026 (2019).

[13] Stringhini, G., Kruegel, C., Vigna, G.: Shady Paths: Leveraging Surfing Crowds to Detect Malicious Web Pages, Proceedings of the 2013 ACM SIGSAC Conference on Computer; Communications Security (CCS'13), pp.133–144 (2013).

[14] 今村祐太，折戸凜太郎，Kritsana Chaikaew，Ceria Manardo，Pattara Leelaprute，佐藤将也，山内利宏：Android における WebView の Web アクセス観測機構を利用した悪性 Web サイトの脅威分析と対策の提案，コンピュータセキュリティシンポジウム 2018（CSS2018）論文集，pp.137-144 (2018).

[15] Imamura, Y., Uekawa, H., Ishihara, Y., Sato, M., Yamauchi, T.:Web Access Monitoring Mechanism for Android WebView, Proceedings of the Australasian Information Security Conference (AISC 2018), pp.1–8 (2018).

[16] 今村祐太，上川先之，石原靖弘，佐藤将也，山内利宏：Android における WebView の Web アクセス観測機構，コンピュータセキュリティシンポジウム 2017（CSS2017）論文集，Vol.2017，No.2，pp.545–552(2017).

[17] 大塚亜未，藤原康宏，村山優子，青柳龍也：スマートフォン利用時の不快感を用いた危険な Web サイトに対する警告インタフェースの実装，マルチメディア，分散，協調とモバイル (DICOMO2019) シンポジウム論文集，pp.923–927 (2019).

[18] Google Developers, Make your web pages fast on all devices, PageSpeed Insights, 入手先 ⟨https://developers.google.com/speed/pagespeed/insights/⟩ (参照 2019–08–08).

[19] 石原　聖，折戸凜太郎，佐藤将也，山内利宏：モバイル向け悪性 Web サイトの探索によるブラックリスト構築手法，コンピュータセキュリティシンポジウム 2019 (CSS2019)，(2019).