4D-02

短縮 URL のオンデマンド安全性検査*

中村章人† 松尾卓朗† 西川直登†

会津大学†

1 はじめに

短縮 URL は､インターネット利用者の利便性を高める便利なしくみである。しかし、その最終的なアクセス先が利用者から隠蔽されてしまうため、フィッシングやマルウェア感染を目的とした悪意あるサイトへの誘導など、セキュリティ上のリスクがある[1,2]。

本論文では、ユーザビリティを損ねることなしに､短縮 URL を安全に利用するしくみを提案する。利用者が短縮 URL をクリックすると､オンデマンドで最終アクセス先リソースの安全性を検査し、危険性が認められた場合のみ検査結果を提示して警告する。検査はブラウザ内のプロキシと Web サービスとの連携で実現する。

2 短縮 URL と URL 短縮サービス

短縮URLとは、あるURL（元URL）に対してその長さが短くなるように変換したURLである。短いURLは、SNS/SMSなどで文章の長さが制限されている場合に文字数の消費を抑えられる、Webページの見栄えを損ねないなどの利点がある。以下に、短縮URLの例を示す。

元URL: https://en.wikipedia.org/wiki/Japan
短縮URL 例1: http://bit.ly/1LXn5Yl
短縮URL 例2: https://goo.gl/J0HsVM

URL短縮システムとは、元URLから短縮URLを生成し、短縮URLから元URLへの逆変換を行うシステムである。主な構成要素は、短縮URLの作成アルゴリズムと、短縮URLと元URLとの対応表である。このシステムをWebサービス化したものをURL短縮サービスという。

短縮URLは、URL短縮サービスのドメイン名と、元URLに対応する一意なキーとで構成される。例えば上の例1では、bit.lyがドメイン名、1LXn5Ylがキーである。同一の元URLに対して、生成される短縮URLはサービスごとに異なる（例1と例２は同一の元URLから生成）。短縮URLから元URLへの逆変換は、その短縮を行ったサービスだけが行える。

上記の例に示したように、短縮URLから元URLを推測することは困難である。この性質を悪用すると、利用者を意図しない悪性リソースへ誘導する攻撃が可能になる[1,2]。

3 URL のオンデマンド安全性検査

我々が提案するURLの安全性検査手法について述べる。既存のURL検査システム[3,4]では、利用者は検査用Webページを開き、対象のURLをタイピングやコピー＆ペーストで入力しなければならない。我々のシステムでは、利用者は通常の操作（画面上のURLをクリックする）以外に追加の操作を必要としない。

3.1 安全性検査の手順とシステム構成

以下に URL 安全性検査の手順を示す（図 1）。

URL 安全性検査の手順:

（１）利用者がブラウザ上でURLをクリックしたら、URL安全性検査クライアント*C*はブラウザに代わってHTTP要求を送信せずにそのURL（短縮URL）URL_SをURL安全性検査サーバ*S*に送る。

（２）URL安全性検査サーバ*S*は、キャッシュにURL_Sの検査結果$R(URL_S)$が存在するならば、その検査結果を*C*に送り、以下の処理を省略する。URL_Sのドメイン名に対応する外部URL短縮サービスを呼び出し、伸長結果の元URL URL_Lを得る。URL_Sが多重に短縮されている場合は、伸張処理を繰り返す。

（３）*S*はURL_Lの安全性検査を外部の検査サービス（設定により複数）に委譲し、それらの結果を統合して検査結果$R(URL_S)$を作成する。

（４）*S*は$R(URL_S)$をキャッシュに保存する。

（５）*S*は$R(URL_S)$をクライアントに送る。

（６）*C*は、サーバから検査結果$R(URL_S)$を受け取り、危険性があると判定されている場合にはそれを警告画面（ウィンドウやタブ、またはポップアップ）に表示する。

---

* “On-Demand Safety Check on Short URLs”, Akihito NAKAMURA, Takurou MATSUO, Naoto NISHIKAWA
† University of Aizu

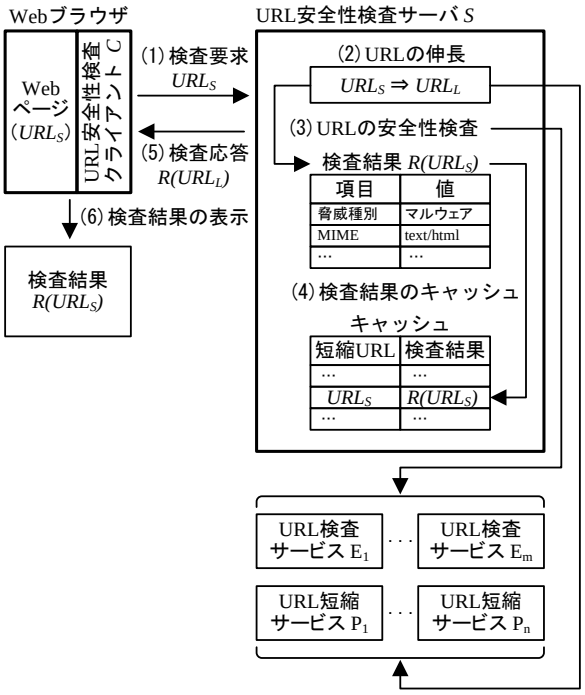

図 1: URL 安全性検査の手順とシステム構成

3.2 検査結果

URL 安全性検査サーバがクライアントに返す検査結果は表 1 の項目から構成される。

表 1: URL 安全性検査結果の内容

短縮 URL 情報: 短縮 URL、元 URL、作成日時、多重短縮の場合に限り伸長過程（短縮 URL と元 URL のリスト）
元 URL 情報: 脅威の種別（マルウェア、フィッシングなど）、MIME タイプ、文字コード、タイトルなどのコンテンツ情報、サムネール画像
レピュテーション情報: リファラ、アクセス数（伸長回数）、アクセス元地域

これらの項目以外にも、セキュリティベンダーが提供する Web セキュリティ評価サービスの結果や悪性サイトのブラックリストなどの情報を統合する予定である。

3.3 システムの実装

提案手法に基づくシステムの基本的な機能は実装済みである。実行環境は、利用者が多いと思われるブラウザChromeを対象とした。他の主要ブラウザへの対応は今後進めていく。サーバ側はRuby on Railsを使用してRuby言語で、クライアント側はChrome Extensions[7]を使用してJavaScriptで開発した。後者は利用者によるインストールが必要である。

サーバのURL安全性検査機能はRESTスタイルのWeb APIからの呼び出しが可能で、異なる実装形態のクライアントやまったく別のアプリケーションの実装にも利用できるようにした。

外部のURL短縮サービスにはBitly、Google、HootSuite、TinyURLを利用し、URL検査サービスにはGoogle Safe Browsing APIs (v4)[5, 6]を用いた。

4 おわりに

本論文では、短縮URLのリスクに着目し、オンデマンドで最終アクセス先リソースの安全性を検査するしくみを示した。クライアント・サーバ構成のシステムを実装済みで、Chromeなどの一般的なブラウザで動作する。利用者は通常のリンククリック以外の追加操作を必要とせず、透過的に安全性検査を実行できる利便性の高さが特徴である。

今後は、検査手法とシステム実装の改善を図るとともに、検査結果の表示のしかたの工夫、およびサーバログの分析に取り組みたい。

5 参考文献

[1] Akhawe, D., Felt, A.P. (2013): “Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness”, 22nd USENIX Security Symposium, 2013.

[2] 中村, 他: “短縮 URL の安全な利用に向けて”, 社会情報学会（SSI）学会大会, 2016.

[3] ExpandURL, http://www.expandurl.net/

[4] 短縮 URL チェッカー, http://x-1.jp/

[5] Google 透明性レポート セーフブラウジングのサイトステータス, https://www.google.com/transparencyreport/safebrowsing/diagnostic/

[6] Google Safe Browsing APIs (v4), https://developers.google.com/safe-browsing/v4/

[7] Google Chrome Extensions, https://developer.chrome.com/extensions