Computer Security Symposium 2017
23 - 25 October 2017

ユーザブロック機能の光と陰: ソーシャルアカウントを特定するサイドチャネルの構成

渡邉 卓弥[1,a)]　塩治 榮太朗[1]　秋山 満昭[1]　笹岡 京斗[2]　八木 毅[1]　森 達哉[2]

概要：本研究は，攻撃者のウェブサイトに訪問したユーザのソーシャルアカウントを特定するサイドチャネル攻撃を実証する．攻撃の骨子となるのは，ソーシャルウェブサービスにおいてユーザをブロックすることで，閲覧できるコンテンツを制御する**ユーザブロック機能**の特性である．攻撃者は事前にアカウントを用意し，サービス上のユーザに対してブロック/非ブロックの状態を任意に設定できる．これを複数組み合わせることで一意のビット列を形成し，大量のユーザに割り当てる．その後ウェブサイトに訪問したユーザに対して，サイト間を跨いだタイミング攻撃によって各アカウントからブロックされているか否かを推定し，ビット列を復元してソーシャルアカウントと紐付ける．検証の結果，SNS，ゲーム，オークション，アダルトサイトなど著名な 12 サービスでアカウントを特定可能であることが明らかとなった．フィールド実験では，最短 4 秒程度という所要時間の下，ほぼ 100%の精度で攻撃が成功することを示した．本稿は，ユーザブロックを逆手にとってユーザのプライバシを脅かすという，現代的なウェブサービスのソーシャル性に起因した新しいセキュリティ問題を提起するものであり，その攻撃原理や対策方法について論じる．

キーワード：システム，ソーシャルアカウント特定，サイドチャネル攻撃，Web セキュリティ

1. はじめに

人と人との相互性に基づいてコンテンツを形成する**ソーシャルウェブサービス**は，登場以来めざましい発展を続け，今日では我々の生活に不可欠な存在となった．その計り知れない影響力は主要サービスにおけるユーザ数の多さだけでなく，個々人が利用するサービス数にも顕れており，Brandwatch [1] の報告によると，インターネットユーザは平均して 5 個以上のソーシャルアカウントを所有している．

ソーシャル機能を持つサービスの種別は多岐に渡るため，アカウントもまた多種多様なプライバシー情報と紐付いている．SNS にはユーザの実名や顔写真，位置情報などが記載されており，オークションサイトの購入履歴 [2] は，ユーザの経済状況や健康状態を反映する．マイクロブログやオンラインゲームのアクティビティは時としてユーザの裏の顔を暴き，出会い系やアダルトサイトは，知られざる交友関係や性的嗜好に関連する．期せずして自身のソーシャルアカウントを特定されることは，インターネットユーザにとってセンシティブなプライバシーの露見につながる．

本研究は，ウェブサイトに訪問したユーザのソーシャルアカウントを特定するサイドチャネル攻撃を実証し，現行サービスが内包する新たなプライバシー漏洩の脅威を明らかにする．攻撃の骨子となるのは，他のユーザに対して自身のコンテンツを表示させないよう制御する**ユーザブロック機能**である．ユーザブロックは，正当なユーザを荒らしや誹謗中傷，スパム行為，ネットストーキングなどから保護する重要な機能としてさまざまなサービスに採用されている．本機能の制限や撤廃はユーザの要望に反する施策であり，アンケート調査 [3] では 90%以上のユーザがブロック数に上限を設けるべきでないと回答した．後に 6 章で考察するように，事業者は対策とサービス品質のトレードオフを勘案しながら，慎重なシステム設計を行う必要がある．

本手法では，攻撃者はユーザブロックによって独自のサイドチャネルを構成し，ユーザごとに一意のシグナルを漏洩させてアカウントを特定する．具体的には，予めアカウントを用意し，サービス上の任意のユーザに対してブロック/非ブロックの 2 つの状態，すなわち 1 ビット情報を自在に設定する．我々はこの特性を**可視性制御**と名付け，攻撃成立のための最も重要な原理として位置付ける．これを複数組み合わせ，大量のソーシャルアカウントに対して一意のビット列を割り当てる．特定対象とするアカウントの数と，攻撃者が用意すべきアカウントの数は二進対数の関係にあり，たとえば 1000 万人をターゲットとするために必

[1] NTT セキュアプラットフォーム研究所
[2] 早稲田大学
[a)] watanabe.takuya@lab.ntt.co.jp

© 2017 Information Processing Society of Japan

要な攻撃者アカウントはわずか 24 個である．未知のユーザがウェブサイトに訪問してきた際には，レスポンス時間の差異からブロック/非ブロックの状態を推定するクロスサイトタイミング攻撃 [4] によってビット列を復元できる．

攻撃の影響範囲を調査するため，世界的に著名な 16 個のソーシャルウェブサービスを検証した結果，Facebook や Twitter，eBay，XVideos など計 12 サービスにおいて攻撃が実現することが示された．さらに実アカウントを用いたフィールド実験では，ほぼ 100%の精度でアカウント特定が成功し，所要時間は，良好な環境においては 4–8 秒程度，遅延を含む劣悪な環境においては 20–98 秒程度であった．

本来ユーザを保護するための機構であるユーザブロックを攻撃に転用するというアイデアは，現代的なウェブサービスのソーシャル性に起因した新たなセキュリティ上の問題を提起する．本稿では，攻撃の背景，具体的な実装方法や実証実験の結果について述べ，さらに攻撃の拡張，制限事項，対策，新規性について論じる．

2. 攻撃の全体像

本章では，ソーシャルアカウント特定攻撃の全体像を説明する．はじめに本攻撃が想定する脅威モデルを示した後，具体例とともに攻撃の流れを説明する．

2.1 脅威モデル

本攻撃は，攻撃者が用意したウェブサイトに訪問したユーザのソーシャルアカウントを特定することを目標とする．脅威となる第一のシナリオは，攻撃者が未知のユーザの実名や顔写真，位置情報といった身辺情報を取得することである．第二のシナリオは，非実名アカウントや，オークションサイト，出会い系，アダルトサイトなど，ユーザが秘匿したいアカウントを特定することである．いずれのシナリオも，ユーザの合意なしに身元やアクティビティが漏洩してしまうため重大なプライバシー侵害を招き，デバイスを跨いだユーザトラッキングや，脅迫行為，ソーシャルエンジニアリングなどの被害に発展するおそれがある．攻撃コードは，ウェブサイトに設置して待ち受けるだけでなく，マルバタイジング [5] や SNS のメッセージ，電子メールのリンクを介して誘導するといった方法が考えられる．

攻撃の前提として，ユーザが自身のウェブブラウザでソーシャルウェブサービスにログインしている必要がある．後に 6 章で説明するように，大多数のユーザはこの前提に該当する．その上で，攻撃者はユーザに対してソーシャルウェブサービスへのリクエストを送信させ，ラウンドトリップタイム (RTT) を取得することで，各アカウントからブロックされているか否かの状態を推定する．この手法はクロスサイトタイミング攻撃 [4] と呼ばれ，ウェブブラウザを対象とし，Same-Origin Policy (SOP) を迂回して情報を取得するサイドチャネル攻撃の一種である．

2.2 攻撃の流れ

攻撃の全体像を図 1 に示す．本手法は攻撃の準備を行う**サイドチャネル制御フェーズ**と，特定を実行する**サイドチャネル復元フェーズ**に分かれる．本章は攻撃の概要を説明することを目的とし，技術的な詳細は 4 章で説明する．

I. サイドチャネル制御フェーズ

本フェーズではユーザブロックを介した**可視性制御**によって，アカウントごとに一意のシグナルを漏洩させるサイドチャネルを構成する．このフェーズは，攻撃実行前に一度だけ行っておけばよい．

ステップ 1. ターゲット列挙：まず攻撃者は，あるソーシャルウェブサービスの中から，特定候補となる N 人のターゲットアカウントを列挙する．列挙の方法としては，引用 [6,7] のような方法を用いて大規模なユーザを網羅したリストを用意するか，あるいは著名人や政治家など，ターゲットとしたい一部のユーザに特化したリストを用意する．

図中では，$N = 8$ の小規模なユーザを列挙した例を示した．攻撃が成立すると，未知のユーザがサイトに訪れた際，8 人のうちどのユーザに該当するかを識別できる．

ステップ 2. ビット割当：次に攻撃者は，ソーシャルウェブサービス上で $2^m \geq N$ を満たす m 個のアカウントを作成する．これらを**シグナルアカウント**と名付け，$S_i, i = 1 \ldots m$ と表す．攻撃者は，各ターゲットに一意のビット列を割り当てた後．ビット列の i 番目の値を，S_i によるブロック (1) あるいは非ブロック (0) に対応付ける．このとき攻撃者は最大で 2^m 人のユーザをターゲットとして列挙できる．なお，後に 4.3 節で示す通り，m を増加させて冗長ビットを追加することで，誤り訂正符号を適用することができる．

図中では，$m = 3$ のときに，8 人のユーザに対して長さ m のビット列を割り当て，各桁の値と S_1，S_2，S_3 のブロック/非ブロックを対応付ける例を示した．

ステップ 3. ユーザブロック実行：攻撃者はシグナルアカウントを制御し，対応表に従ってユーザをブロックする．

図中では，攻撃者はシグナルアカウント S_1 から Erin，Frank，Grace，Heidi をブロックし，以下同様に S_2，S_3 からのブロックを対応表にしたがって実行していく．

II. サイドチャネル復元フェーズ

本フェーズでは，クロスサイトタイミング攻撃によって未知のユーザのブロック/非ブロック状態を復元し，ソーシャルアカウントの特定を行う．このフェーズは，ユーザが攻撃者のウェブサイトにアクセスするたびに実行する．

ステップ 1. ユーザ訪問：ユーザが攻撃者の制御するウェブサイトにアクセスした時に，アカウント特定を開始する．ウェブサイトには攻撃を実行するための JavaScript コードが含まれており，ユーザのブラウザ上で実行される．

ステップ 2. RTT 測定：ユーザが実行したコードは，各シ

© 2017 Information Processing Society of Japan

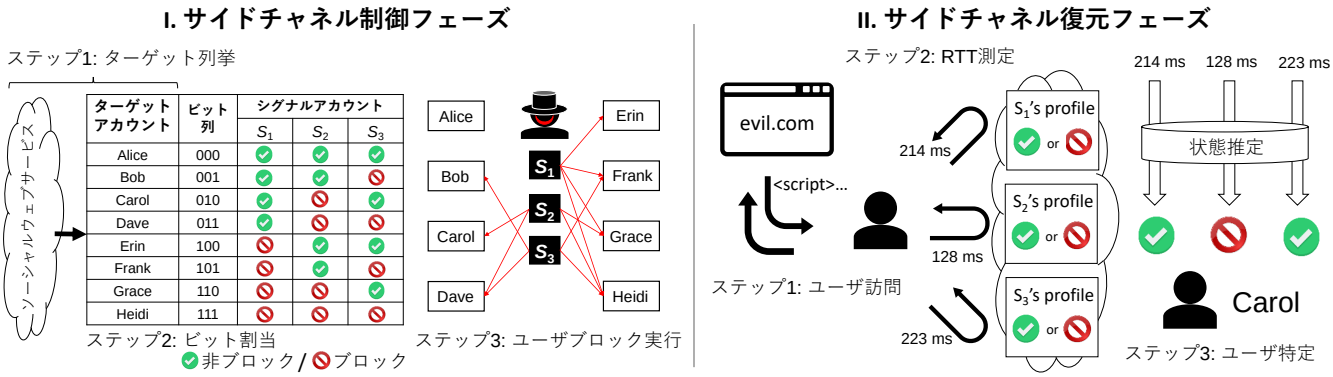

図 1 攻撃の全体像

グナルアカウントのプロフィールページへ GET リクエストを秘密裏に送信する．攻撃者は SOP の制約で通信の中身を見ることができないが，代わりにリクエストの所要時間 (RTT) を測定できる．測定を反復して行うことで，所要時間増加と引き換えに推定精度をより高めることができる．

図中では，ユーザに S_1，S_2，S_3 の各プロフィールページへリクエストを送信させた．簡単のため，ここでは測定を一度ずつ行っている．計測された RTT は，それぞれ 214 ms，128 ms，223 ms であった．

ステップ 3. ユーザ特定：最後に，攻撃者は計測した RTT からブロック状態を識別し，アカウントを特定する．ブロック/非ブロックに応じて RTT に統計的差異が発生するため，各ビット桁を推定できる．攻撃者は復元したビット列を対応表と照合し，ユーザを特定する．

図中では，214，128，223 ms について，非ブロック，ブロック，非ブロックと推定した．これはビット列 {101} として復元できるため，攻撃者はこのユーザが Carol であると判別できる．具体的な推定手順は 4 章で詳述する．

3. ユーザブロックによるサイドチャネル

本章では，ユーザブロックを用いたサイドチャネル攻撃の実現可能性を示す．まず，ユーザブロック機能の技術概要について説明する．続いて，ブロック/非ブロック時における各リクエストの RTT の特性を調査する．最後に，世界的に著名な 16 のソーシャルウェブサービスについて，RTT を統計的に識別できるか否かを実験する．

3.1 ユーザブロック機能

ユーザブロックは，正当なユーザが悪質なユーザに対してコンテンツ閲覧を禁止，すなわち**可視性制御**することでハラスメントから身を守る機能と言い換えることができる．ここで忘れてはならないのは，悪質なユーザもまた，正当なユーザの可視性を制御できてしまうという事実である．

ソーシャルウェブサービスは，ユーザが他者からブロックされているか否かに応じて表示が異なるページを有しており，典型的にはユーザプロフィールが挙げられる．

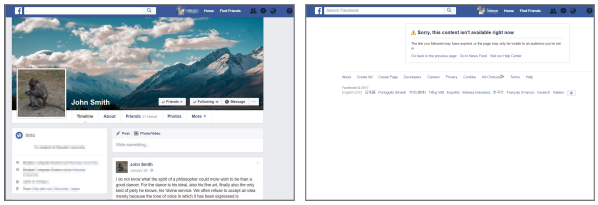

図 2 平常時 (左) とブロックされた時 (右) のページの差異

Facebook における，平常時とブロックされている時のページの違いを図 2 に示す．平常時ではユーザの自己紹介文や写真，最新の投稿などが表示されるが，ブロック時にはそれらの情報が遮断される．プロフィール以外にも，eBay の商品落札ページや，Tumblr の投稿を格納した JSON など，他のコンテンツが遮断されるケースが存在する．

ユーザをブロックする API は必ずしも提供されていないが，ヘッドレスブラウザや，ブロック時の通信内容を再現するスクリプトによって大規模なブロックを自動化できる．現状ブロック上限が明記されているサービスは少なく，我々の予備実験では Facebook，Twitter，Tumblr といったサービスで 300 万以上のユーザをブロックできた．

我々が行ったユーザブロックに関する利用統計 [3] では，1000 以上のアカウントをブロックするユーザが 1.7%存在した．分母の大きさを考えると無視できない数であり，発言数やフレンド数が多いヘビーユーザの割合が高い．さらに事前のアンケート調査では，Twitter/Facebook ユーザの 52.3%/41.4%がブロック機能を利用していると回答し，92.4%/93.9%が上限を設けるべきでないと回答した．本機能の制限や撤廃はユーザの要望に背く施策であり，サービスの品質と健全性に暗い影を落とす．加えて，ブロック上限による対策は 4.3 節に示すような攻撃の拡張によって，容易に突破することができる．

3.2 ブロック/非ブロック時の RTT 特性

RTT 特性を調べるため，あるユーザアカウントにログインした状態で，シグナルアカウントのページにリクエストを送信して時間を計測する JavaScript を実行した．以下では代表的なソーシャルウェブサービスとして，Facebook，

© 2017 Information Processing Society of Japan

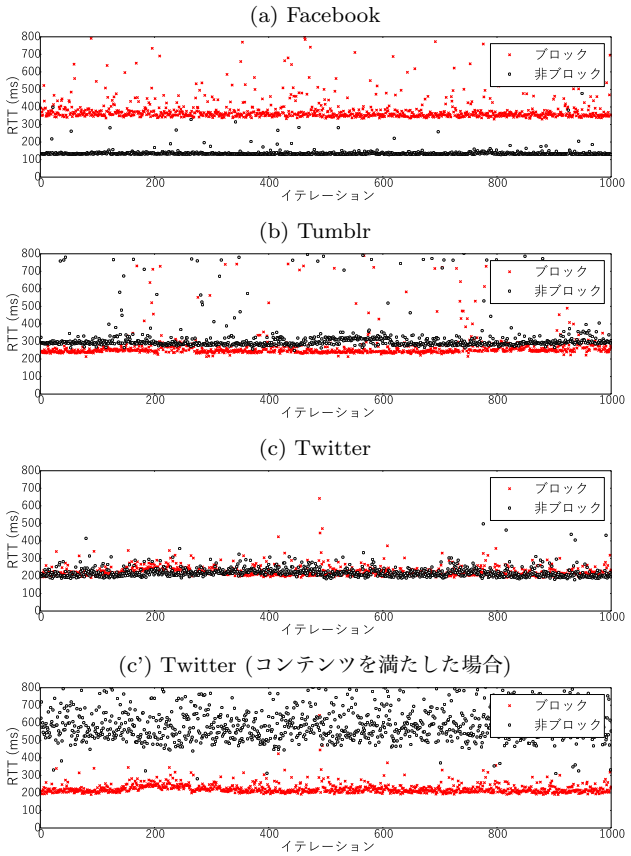

図 3 ブロック時と非ブロック時のページに対する RTT の分布

Tumblr，Twitter の 3 つを用いる．図 3 は，実際に測定した RTT の分布である．(a) Facebook の例では，ブロック/非ブロックそれぞれの分布の間に，明らかなギャップが存在することがわかる．(b) Tumblr では (a) よりも差が小さいものの，やはりギャップが見受けられる．(c) Twitter においては，一見するとブロック/非ブロックの RTT が混在しており，明確なギャップを見出すことができない．しかしこういったケースにおいても，攻撃者はシグナルアカウントのページに可能な限り多くの情報を配置することで，ギャップを発生させることができる．具体的には，自己紹介文や最新の投稿に大量の文字列，リンクなどを記載し，コンテンツサイズを増幅させる．増幅後の分布を (c') に示す．ブロックされていないときには多くのレスポンス時間を要するようになるが，ブロックされているときのコンテンツは遮断されているため，レスポンス時間が変化しない．したがって，ブロック/非ブロックの間に大きなギャップが発生し，識別性を高めることができる．

直感的には，より大きなコンテンツを返す非ブロック時には，ブロック時よりも大きな RTT が計測されると予期される．(b) や (c') はこれを支持する結果となっているが，(a) はブロック時の RTT の方が大きな値を示している．サービスの実装を確認することはできないが，サーバ側のキャッシュなどの機構によって，非ブロック時のレスポンスが高速化している可能性が考えられる．識別のためには，両者の RTT に統計的な差異が存在することが重要であって，その大小関係は識別性に影響を与えない．

3.3 RTT の識別性

最後に，実際のサービスに対してブロック/非ブロック時の RTT を識別できるか検証した．識別には，マン・ホイットニーの U 検定を用いる．U 検定は，2 つの独立したサンプル群の大小関係をもとに p 値を導き，有意差を検定するノンパラメトリック手法である．世界的に著名な 16 のソーシャルウェブサービスでブロック/非ブロック時の RTT を 100 回ずつ計測し，この 2 群を有意水準 0.01 として検定した．結果，少なくとも Facebook，Instagram，Tumblr，Google+，Twitter (SNS)，eBay (オークション)，Medium (フォーラム)，Roblox，Xbox Live (ゲーム)，PornHub，Xvideos (アダルト)，Ashley Madison (出会い) の 12 サービスで識別が可能であった．これらは 5000 万–20 億人のユーザ数を誇るサービスであり，攻撃のインパクトは極めて大きいといえる．一方，Quora，Flickr，DeviantArt，Meetup ではコンテンツ差異が小さく，今回の調査では $p > 0.01$ となった．これらのサービスでも試行回数を増やすことで，統計的有意差が発生する可能性がある．

4. ユーザ特定攻撃

本章ではまず，ユーザ特定攻撃の定式化を行う．続いて，RTT をもとにブロック状態を推定する手法を説明する．最後に，攻撃の精度や実現性を高める 2 つの拡張を提案する．

4.1 定式化

シグナルアカウントの数を m，ターゲットアカウントの数を N とする．m は $2^m \geq N$ を満たす最小の値であり，例えば 100 万人がターゲットならば $m = 20$ を設定する．

サイドチャネル制御フェーズでは，攻撃者はターゲットアカウントに長さ m のビット列を割り当てた対応表を用意する．$U_i \quad (i = 1, \ldots, N)$ をターゲットとしたとき，対応表は各 U_i に対応したビット列 $B_i = \{b_1 b_2 \ldots b_m\}$ を有する．ここで，$b_j \in \{0, 1\}$ は j 番目のビットに対応する．このとき B_i は，U_i をエンコードした値としてみなすことができる．すなわち，$B_i = \mathrm{encode}(U_i)$ とする．

続いてシグナルアカウント $S_j \quad (j = 1, \ldots, m)$ を次のように構成する．B_i の j 番目のビットを b_{ij} としたとき，各 i, j について $b_{ij} = 1$ ならば S_j から U_i をブロックする．各ビットは 0.5 の割合で $b_{ij} = 1$ となるため，シグナルアカウントがブロックすべきアカウントの数は $N/2$ である．

サイドチャネル復元フェーズでは，攻撃者はウェブサイトに訪問したユーザに対し，m 個のシグナルアカウントに対するリクエストを送信させ，RTT を測定する．ここで，S_j について得られた RTT のシーケンスを $\mathbf{R}_j = \{R_1, R_2, \ldots\}$ とする．さらに，$\widehat{b_j} \in \{0, 1\}$ を，$\mathbf{R}_j$ から推定されたブロッ

© 2017 Information Processing Society of Japan

ク/非ブロック (1/0) の値とする．このとき全体のビット列 B を推定した結果を $\widehat{B} = \{\widehat{b_1} \dots \widehat{b_m}\}$. と表すことができる．最後に，サイドチャネル制御フェーズで作成した対応表に従い，ビット列に対応するユーザを特定する．すなわち，推定ユーザ $\widehat{U} = \mathrm{decode}(\widehat{B})$ となる．

4.2 ブロック状態の推定

はじめに，ウェブサイトに訪問したユーザがターゲットリストに含まれているかについてテストする．これを**メンバーシップテスト**と名付ける．攻撃者はターゲットアカウントをすべてブロックする**クローズアカウント**と，一切ブロックしない**オープンアカウント**の 2 つを用意する．クローズアカウントとオープンアカウントへの RTT を計測し，有意差が見られる場合，ユーザはターゲットリストに含まれていると判定し，攻撃を続行する．そうでない場合，ユーザはクローズアカウントからブロックされていないので，ターゲットリストに含まれていないとみなして攻撃を終了する．誤差の影響を考慮し，測定は k_0 回反復して行う．計測された RTT 間に有意差があるかを判定するためには U 検定を用いる．U 検定の計算は非常に軽量であるため，メンバーシップテストは RTT を測定した直後に完了する．本手法では，有意水準 $\alpha = 0.01$ を採用した．

その後，ユーザが各シグナルアカウントからブロックされているか否か，ビットの値を推定する．先ほどクローズアカウントとオープンアカウントに対して計測した RTT の 5 パーセンタイル値を $C_{0.05}$ および $O_{0.05}$ とする．5 パーセンタイル値は外れ値やノイズの影響を除去するために採用した．これらの値は，状態を推定するための閾値として使用する．S_j に対する RTT を k 回測定し，同様に 5 パーセンタイル値 $R_{0.05j}$ を計算する．$R_{0.05j}$ が $C_{0.05}$ に近い場合，ユーザは S_j にブロックされていると推定し，$O_{0.05}$ に近い場合はブロックされていないと推定する．すなわち，

$$\widehat{b_j} = \begin{cases} 1 & \text{if} \quad |R_{0.05j} - C_{0.05}| < |R_{0.05j} - O_{0.05}|, \\ 0 & \text{else.} \end{cases}$$

上記プロセスをすべての $j \in \{1, \dots, m\}$ に対して繰り返すことで，攻撃者はユーザのビット列 $\widehat{B} = \{\widehat{b_1} \dots \widehat{b_m}\}$ を推定できる．最後に前節の手順で $\widehat{U} = \mathrm{decode}(\widehat{B})$ とし，ビット列からターゲットアカウントを復元する．

4.3 拡張

本節では，攻撃の成功率をより高めるための**誤り訂正符号**と，ユーザブロック数に上限がある際にターゲット数を拡大する**ユーザ空間分割**の 2 つの拡張について説明する

誤り訂正符号：安定した環境下ではブロック/非ブロック時の RTT に明確な差異が発生するため，前述の手法で高精度に状態を推定できる．一方で，突発的なサーバエラーのような異常値がビット誤りを招く可能性がある．識別をより確実なものとするために，誤り訂正符号を適用できる．本稿では，訂正能力が高く実装が容易であるリードソロモン符号を採用した．より高い性能を求める場合には，誤りの分布などの特性に応じて，他のアルゴリズムを選択することも可能である．今回は概念実証を目的とするため，最適な誤り訂正アルゴリズムの検討はフォーカス外とする．

リードソロモン符号は，冗長シンボルの数を K，シンボルの大きさを r ビットとすると，$K/2$ までのシンボル誤りを訂正できる．ユーザに割り当てた元のビット列は大きさ m なので，用意する必要があるシグナルアカウントの数は $m + rK$ 個である．サイドチャネル制御フェーズでは，ターゲットに割り当てたビット列をエンコードし，新たに生成されたビット列にしたがってブロックを実行する．サイドチャネル復元フェーズでは，クロスサイトタイミング攻撃によって得られたビット列をデコードすることで，誤り訂正後のビット列を得ることができる．

ユーザ空間分割： 3 章で論じた通り，ブロック機能に制限を課す対策はサービスの品質を著しく低下させるおそれがある．それでもなお，本攻撃を危惧して上限を設けるサービスに対しては，以下の手順が有効である．今まで説明した手法では，ブロック上限を L としたとき，クローズアカウントがブロックできる数 (すなわちターゲット数の上限) も L となる．ここでターゲットリストを S 個に分割することで，LS 人まで網羅することができるようになる．

サイドチャネル制御フェーズでは，各ユーザ空間 $j \in \{1, \dots, S\}$ に対し，j 番目の空間に属する L 人のターゲットをすべてブロックしたクローズアカウントをそれぞれ用意する．さらに空間ごとに $\lceil \log_2 L \rceil$ 個ずつシグナルアカウントを用意し，空間内の各ターゲットにビット列を割り当てブロック/非ブロックを設定する．最終的に用意すべきシグナルアカウントの数は $S\lceil \log_2 L \rceil$，クローズアカウントは S，オープンアカウントは 1 つである．

サイドチャネル復元フェーズでは，まず (1) ユーザが属する空間を識別し，その後 (2) 空間内のどのターゲットであるかを特定する．(1) では，1 つのオープンアカウントと S 個のクローズアカウントに対して RTT を計測し，U 検定を繰り返してユーザ空間を決定する．この手順は前節におけるメンバーシップテストの拡張といえる．(2) では，$\lceil \log_2 L \rceil$ 個のシグナルアカウントに対して RTT を計測して状態を推定し，ビット列を復元する．空間ごとに異なるシグナルアカウントを使用するため，(1) の結果に応じてリクエスト先を分岐させる必要があることに注意する．推定に使う閾値としては，(1) で得られた RTT を活用できる．

5. フィールド実験

本章では， 3 章で異なる RTT 特性を示した Facebook,

© 2017 Information Processing Society of Japan

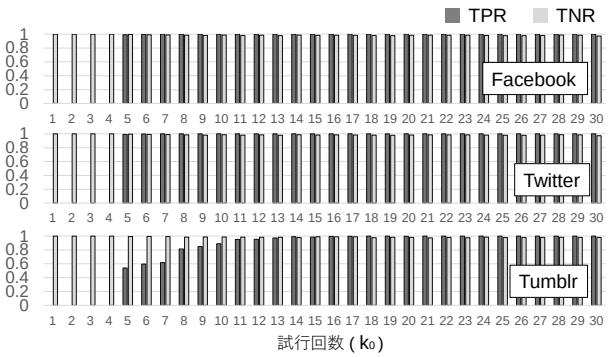

図 4 試行回数 k_0 と TPR/TNR の関係

Twitter，Tumblr についてフィールド実験を行い，実現可能性を示す．まず 5.1 節で，RTT からブロック状態を推定する精度を評価する．続いて 5.2 節で，ユーザ特定の成功率を示す．最後に 5.3 節で攻撃の所要時間について述べる．

5.1 ブロック状態の推定精度

この実験では，一般のラップトップ PC にインストールされた Google Chrome で，JavaScript によるクロスサイトタイミング攻撃を実行した．紙面の都合により割愛するが，攻撃は IE と Firefox でも成立することを確認している．

メンバーシップテスト：はじめに，オープンアカウントとクローズアカウントに対して k_0 回リクエストを送信し，U 検定を適用するメンバーシップテストを行った．ターゲットリストに含まれるアカウントと含まれないアカウントにログインした状態で，100 回繰り返しテストする．ユーザがターゲットに含まれることを正しく判定した比率を True Positive Rate (TPR)，含まれないことを正しく判定した比率を True Negative Rate (TNR) とした．

k_0 と TPR/TNR の関係を図 4 に示す．TNR はすべての k_0 で 0.97 以上であったが，TPR は k_0 が小さいときに低い値を示した．これはブロック/非ブロック時の有意差を判定するために必要なサンプル数が得られていないためである．しかし，k_0 を増加させていくことで TPR の値は 1.0 に漸近した．本実験では，すべてのサービスで高い精度が得られる $k_0 = 30$ を選択した．ここで得た $O_{0.05}$ と $C_{0.05}$ をビット推定の閾値として活用する．

ビット値推定：次に，シグナルアカウントのブロック/非ブロックの状態，すなわち各ビット値を推定する精度を評価した．各状態について RTT 計測を k 回行って 5 パーセンタイル値を導出し，$O_{0.05}$ と $C_{0.05}$ のいずれに近いかによって状態を推定する．ブロック/非ブロックの状態を正しく推定した比率を，それぞれ True Block/Non-Block Rate (TBR/TNBR) とした．実験結果を表 1 に示す．全体的に高い精度を示しており，Facebook では，$k = 3$ で一度も誤ることなく推定に成功した．$k \geq 20$ で，すべてのサービスにおいて推定が完璧に成功した．

表 1 ビット値推定の精度

	Facebook		Twitter		Tumblr	
k	TBR	TNBR	TBR	TNBR	TBR	TNBR
1	1.00	0.98	0.99	0.99	0.67	0.99
3	1.00	1.00	1.00	0.99	0.89	0.99
5	1.00	1.00	1.00	0.97	0.95	0.98
10	1.00	1.00	1.00	1.00	0.98	1.00
20	1.00	1.00	1.00	1.00	1.00	1.00
30	1.00	1.00	1.00	1.00	1.00	1.00

表 2 ソーシャルアカウント特定攻撃の成功率

	Facebook/有線	Twitter/Wi-Fi	Tumblr/テザリング
TNR	1.00 (20/20)	1.00 (20/20)	0.95 (19/20)
TPR	1.00 (20/20)	1.00 (20/20)	1.00 (20/20)
IDR	0.95 (19/20)	1.00 (20/20)	1.00 (20/20)
IDR/EC	1.00 (20/20)	1.00 (20/20)	1.00 (20/20)

5.2 攻撃の成功率

本節では，実際の攻撃シナリオを模した環境における成功率を示す．ビット列の長さは $m = 24$ とし，さらにブロック長 4 ビットの冗長ブロックを 2 つ設定した．リードソロモン符号によって，1 ブロックまで誤りを訂正できる．以上，32 のシグナルアカウントとオープンアカウント，クローズアカウントを含む 34 個のアカウントを用意し，ターゲットリストに対して適切なユーザブロックを実行する．

ターゲットとして，我々が実際に使用している 10 個のソーシャルアカウントを列挙し，長さ 24 のビット列をランダムに割り当てた．割り当てたビット列をリードソロモン符号でエンコードし，長さ 8 の冗長ビットを導出した．加えて，ターゲットに含まれないアカウントを別途 10 個用意した．以上，計 20 個のアカウントそれぞれにログインした状態で，攻撃者ウェブサイトにアクセスを行い，アカウントを正しく特定できるかどうかを判定する．ウェブサイトへの訪問は各アカウントで 2 回ずつ，合計で 40 回実施する．さまざまなネットワーク環境で実現することを示すために，有線 LAN，Wi-Fi，テザリングの三種類を用意した．Facebook/有線 LAN，Twitter/Wi-Fi，Tumblr/テザリングの組み合わせで実験を行い，パラメータは $k = 30$ とする．前節と同様に，ターゲットリストに含まれている/含まれていないユーザを正しく識別した比率を TPR/TNR とする．さらに，特定に成功した比率を Identified Rate(IDR) として示し，誤り訂正込みで成功した比率は IDR/EC と表記する．以上の実験結果を，表 2 にまとめた．

前節の結果からも予期できた通り，総じて非常に高い精度で特定に成功した．以下では失敗したケースについて考察する．1 つ目の失敗は，Facebook におけるアカウント特定の失敗である．ネットワークログによると，このとき 1 秒程度の間，502 サーバエラーが発生していた．エラーによって平常時の 1/5 程度の RTT が何度か計測されてしまい，1 ビットの誤りが発生した．しかしながら，リードソロモン符号を適用した場合にはこのビットは修復され，

© 2017 Information Processing Society of Japan

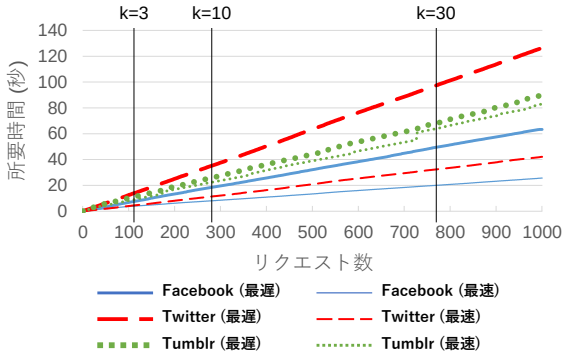

図 5 リクエスト数と所要時間の関係

ユーザを正しく特定することに成功した．2 つ目の失敗は，Tumblr において，ターゲットリストに含まれないユーザをターゲットとみなしてしまった例である．これは，ユーザがオープンアカウントからもクローズアカウントからもブロックされていないにも関わらず，$p < 0.01$ となる RTT を偶然計測してしまったケースである．ここで誤ってビット推定の段階まで進んだ際に，再びリードソロモン符号が効力を発揮する．すなわち，誤り箇所が多すぎる場合に検出に失敗するという性質を利用して，攻撃側はメンバーシップテストが失敗した可能性があると判断し，もう一度攻撃をやり直すという選択を取ることができる．

5.3 攻撃の所要時間

所要時間が短いほど，攻撃は成立しやすいものとなる．所要リクエスト数は計算によって求められるが，リクエストを完了するために必要な時間は RTT に依存する．リクエスト数と所要時間の関係を実験によって導出し，その結果を図 5 に示した．図中における「最速」は，すべてのビットをブロック/非ブロックのうち RTT が小さい方に割り当てられたユーザを想定し，「最遅」はその反対のユーザを想定する．リクエストの並列数は，主要ブラウザにおけるデフォルトの最大同時接続数である 6 を設定した．

前節までと同様に $k_0 = 30$ に統一すると，m ビットの推定，すなわち 2^m 人のターゲットの中からユーザを識別するために必要なリクエストの総数は $mk + 2 * 30$ である．たとえば $k = 30, m = 24$ とすると，リクエスト総数は 780 となる．このとき Facebook では 20–50 秒，Twitter では 32–98 秒，Tumblr では 64–68 秒で特定が完了する．表 1 によれば，Twitter においては $k = 10$ で TBR/TNBR ともに 1.0 を示した．このときのリクエスト数は 300 であり，所要時間は 12–37 秒である．さらに Facebook 上では，$k = 3$ としても十分に高い精度が得られる．リクエスト総数は 132 であり，わずか 4–8 秒で攻撃が完了する．

6. 考察

6.1 制限事項

本攻撃はユーザがサービスにログインしているという前提，つまりブラウザの Cookie 保持状態に依存する．概して，ソーシャルウェブサービスにおける Cookie の有効期限は長く [8]，数年単位で設定されている場合もある．これはサービスが集客や広告表示のために，あえてログアウトしづらいような仕組みを取っているためである．ユーザにとっても，ソーシャルプラグインを利用したり，認証情報を入力する手間を省略できる利点がある．また本攻撃は，プライベートブラウズを利用するユーザには有効でない可能性がある．しかし Bursztein [9] によれば，現状プライベートブラウズを利用したことのあるユーザはわずか 20%に留まっており，それらのユーザも限られた用途でのみ利用することがわかっている．また上述した収益性および利便性の観点から，サービス側からは忌避される機能である．

クロスサイトタイミング攻撃は，Android や iOS 端末上のブラウザでも動作するが，モバイル独自のエコシステムが本攻撃の制約となりうる．ソーシャルプラグインやソーシャルログインによって，依然としてモバイルブラウザでログインする機会は存在するものの，多くのサービスの利用形態はアプリをベースとしたものとなっている．このとき，サイト間ではなくアプリ間で情報を窃取するサイドチャネル攻撃 [10] が有効な可能性がある．モバイルへの拡張性に関するさらなる調査は今後の課題としたい．

6.2 対策方法

サービスは，レスポンス時間を調整することで，ブロック/非ブロック時の RTT 差異を最小限に抑えることができる．具体的には，レスポンス時間が短い方にあえて遅延を加える方法や，全体にランダム遅延を加える方法がある．しかしいずれの方法も，性能の劣化からユーザエクスペリエンス (UX) に無視できない悪影響を及ぼしてしまう．また，ネットワーク上の遅延はサービスから制御できないため，時間差を完全に抑制することはできない．加えて，サイドチャネル分野における攻防の歴史から，この種の対策は統計的なアプローチ [11] やキャッシュベースの攻撃 [12] によって回避されてしまうことがわかっている．

CSRF を防ぐ手法であるトークン検証は，クロスサイトタイミング攻撃に対しても有効である．サーバは遷移前のページで一時的なトークンを発行し，Cookie や URL リンクに付与する．その後，発行したトークンを保護したいページで検証し，一致しない，あるいは付与されていない場合はアクセスを拒否する．本対策の欠点は，正常なアクセスにも適用されうる点である．SNS のプロフィールは検索エンジンなどから直接リンクされる機会が多いため，そういったアクセスの棄却は利便性の妨げになるおそれがある．そこで，トークンなしでアクセスできる代替ページを用意し，後から XMLHttpRequest でコンテンツを取得する方法ならば，正常な直接リンクを機能させたまま CSRF を防止できる．ただし余分な通信や描画を追加する必要が

© 2017 Information Processing Society of Japan

あるために，UX への悪影響は免れないと考えられる．

6.3 研究倫理

本研究では，攻撃の実現性と影響力を評価するため，実サービスを用いた実験を行った．実験は限られた量のトラフィックしか生成しないよう注意深くデザインされ，サービスの運営に悪影響を与えることはなかった．またユーザ特定は我々が保有するアカウントに対してのみ行われ，他のユーザは攻撃に一切関与していない．本研究で提唱された攻撃はサービスの脆弱性に起因するものではないが，実サービスに与えるインパクトを鑑みて，手法の詳細や実験結果を事業者に報告する準備を進めている．

7. 関連研究と新規性

本章では，本研究の関連研究と新規性について述べる．**サイドチャネル復元フェーズ**で用いたクロスサイトタイミング攻撃は，Bortz ら [4] のアイデアをもとに開発した．これは外部にリクエストを送信させてから処理が終わるまでの時間を計測するシンプルな手法であるが，ゆえに対策が困難であり，発表から 10 年経った今日でもあらゆるサービス・ブラウザで動作する．その後，HTML5 の実装によって，キャッシュベースでタイミング攻撃を行う方法 [12] が新たに提唱された．また具体的な適用例として，ウェブメールの内容 [13] や位置情報 [14] を推定するシナリオが考案された．上述した研究すべてに共通しているのは，ユーザ自身の操作や状態に依存したリソース，つまり攻撃者の制御が及ばない情報の窃取を目的とする点である．

したがって，本研究の最大の新規性は，**サイドチャネル制御フェーズ**にあるといえる．ユーザブロックが与える**可視性制御**の特性により，攻撃者はユーザから漏洩するシグナルを自在に操ることができる．結果として，大量のターゲット候補を符号化し，最小限の試行回数でアカウントを一意に特定する独自のサイドチャネルを構成した．ユーザブロックはソーシャルウェブサービスに備えられた一般的な機能であるため，本攻撃の影響範囲は極めて大きい．

また，ソーシャルアカウント特定によってウェブユーザの匿名性を奪う研究として，Wondracek ら [15] はブラウザの訪問履歴からユーザの所属するグループを推定し，アカウントを絞り込めるまで繰り返す手法を実証した．この手法も，やはり漏洩する情報はユーザ依存であり，たとえばいずれのグループにも属していないアカウントを特定することはできない．さらに，この技術は訪問済みリンクのスタイル情報から訪問履歴が漏洩する *history stealing* に基づいていたが，2011 年，CSS の仕様変更 [16] によって対策がなされ，現在ではこの攻撃は成立しない．

8. おわりに

本稿では，攻撃者のウェブサイトに訪問したユーザのソーシャルアカウントを特定する攻撃を実証した．攻撃者はユーザブロック機能を介してコンテンツの可視性を自在に制御し，ユーザを一意に識別できるサイドチャネルを構成する．本攻撃は，現存するさまざまなソーシャルウェブサービスで実現可能であり，標的の環境を問わず高精度かつ高速に動作することを示した．

悪質なユーザの活動を抑制するために設計された機能が，翻って正当なユーザのプライバシーを脅かす—我々の示したアイデアは，現代的なウェブサービスのソーシャル性に根ざす新しい問題点を明らかにした．本研究で得られた知見がシステムの再設計に活用され，また事業者およびユーザ意識の向上につながることを期待する．

参考文献

[1] Brandwatch, “Marketing: 96 Amazing Social Media Statistics and Facts.” `https://goo.gl/XXs3KR`, 2016.

[2] T. Minkus and K. W. Ross, “I Know What You’re Buying: Privacy Breaches on eBay,” in *Proc. of PETS*, 2014.

[3] T. Watanabe, “Survey on User Blocking.” `https://github.com/NWSecLab/user-blocking-survey`.

[4] A. Bortz, D. Boneh, and P. Nandy, “Exposing Private Information by Timing Web Applications,” in *Proc. of WWW*, 2007.

[5] Z. Li, K. Zhang, Y. Xie, F. Yu, and X. Wang, “Knowing Your Enemy: Understanding and Detecting Malicious Web Advertising,” in *Proc. of ACM CCS*, 2012.

[6] B. Krishnamurthy, P. Gill, and M. Arlitt, “A Few Chirps about Twitter,” in *Proc. of WOSN*, 2008.

[7] R. Dey, C. Tang, K. Ross, and N. Saxena, “Estimating Age Privacy Leakage in Online Social Networks,” in *Proc. of INFOCOM*, 2012.

[8] G. Kontaxis, M. Polychronakis, A. D. Keromytis, and E. P. Markatos, “Privacy-Preserving Social Plugins,” in *Proc. of USENIX Security*, 2012.

[9] E. Bursztein, “Understanding how people use private browsing.” `https://www.elie.net/blog/privacy/understanding-how-people-use-private-browsing`.

[10] C. Q. Alfred, Q. Zhiyun, and M. Z. Morley, “Peeking into your app without actually seeing it: Ui state inference and novel android attacks.,” in *Proc. of USENIX Security*, 2014.

[11] P. C. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,” in *Proc. of CRYPTO*, 1996.

[12] T. V. Goethem, W. Joosen, and N. Nikiforakis, “The Clock is Still Ticking: Timing Attacks in the Modern Web,” in *Proc. of ACM CCS*, 2015.

[13] N. Gelernter and A. Herzberg, “Cross-Site Search Attacks,” in *Proc. of ACM CCS*, 2015.

[14] Y. Jia, X. Dong, Z. Liang, and P. Saxena, “I Know Where You’ve Been: Geo-Inference Attacks via the Browser Cache,” *IEEE Internet Computing*, vol. 19, pp. 44–53, 1 2015.

[15] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel, “A Practical Attack to De-anonymize Social Network Users,” in *Proc. of IEEE S&P*, 2010.

[16] W3C, “5.11.2 The link pseudo-classes: :link and :visited.” `https://www.w3.org/TR/CSS2/selector.html\#link-pseudo-classes`, 2011.

© 2017 Information Processing Society of Japan