個人情報及び個人識別子を含むファイルと通信を検出するための双子の環境

張 世申　三村 賢次郎　新城 靖

平成 30 年 10 月 12 日

概 要

インタネットユーザは普段ネットワークサービスを利用する時に、プライバシデータがサービス提供者に収集されることがよくある。機械学習ブームの原因でユーザデータの重要性がより高くなる。サービスの提供者は自分のアプリケーションの体験を向上させるため、ユーザの個人データを分析、販売することがよくある。通常のログインを通じて情報漏洩とは違い、ユーザがログインしなくてもユーザトラッキングの手段でユーザのプライバシが漏れることもよく発生する。これらのユーザの身元を識別できるタグは、広告やオークションのウェブサービスに利用され、ユーザの使用に悪影響をもたらすこともよく発生する。本研究では、ユーザプライバシの保護を目標として、ユーザトラッキングに使われる情報を検出、保護したい。そこで本研究では Docker コンテナ技術を利用し、双子の環境というユーザトラッキングを検出できるアプリケーションの仮想実行環境を提案した。双子の環境を利用する双子のブラウザを実装し、ユーザトラッキングを検出した。

1　はじめに

PC で動作するアプリケーションは、Web ブラウザのように明示的に通信を行うものだけでなく、オフィスツールのように、ユーザの意図しない通信を行うものがある。Web ブラウザであっても、ユーザトラッキングのために暗黙的に通信を行うことがある。このような意図しない通信により住所・氏名等の個人情報、および cookie のように個人情報と結び付けられた個人識別子が送信されることがある。

本研究では、コンテナという OS 層の仮想実行環境を利用し、個人情報及び個人識別子が含まれているファイルと通信を検出することを提案する [4]。そして、ファイルや通信から不要な情報を削除するツールを実装する。個人情報および個人識別子を検出する対象は、次の2 つである。

- ファイル
- ネットワーク通信

たとえば、Web ブラウザは、個人識別子として訪問履歴や、フォームの内容、Cookie などを保存する。ユーザトラッキングでよく利用される Cookie にはサーバが配るユニークな ID やセッション ID が含まれている。しかし、現在の Web ブラウザやオフィスツールは、非常に複雑であり、これらの識別子がどのファイルにどのような形式で保存されているかを調べることは容易ではない。本研究では、双子の環境を実装して、個人情報および個人識別子が含まれているファイルや通信を検出する。なお、以下では、個人情報と記載した時にも、個人情報と個人識別子の両方を含むものとする。

2　双子の環境と双子のブラウザによる個人情報の検出

双子の環境とは、プログラムファイルやデータファイル等の内容がほとんど同じであるような 2 つの仮想実行環境である (図 1)。人間における双子の研究では、異なる双子が異なる環境で育てられた際にそれぞれの医学的、遺伝子的、心理学的性格を調査ことで、どのような違いが生まれるかを調査する場合が多い。本研究で提案する双子の環境では、類似の 2 つの環境で同一のプログラムをそれぞれ実行し、同一の入力を与える。そして、2 つのプログラムの動作上の相違点を検出する。

本研究室では、双子の環境で動作する Web ブラウザとして、双子のブラウザを開発している [3]。双子のブラウザとは、双子の環境で協調動作する 2 つのブラウザである。

本研究では、双子のブラウザを用いてサーバによるユーザトラッキングを検出する。ユーザトラッキングの手法としては、Cookie を使う方法や URL にタグを埋め込む方法がある。それ以外に、Flash Cookie

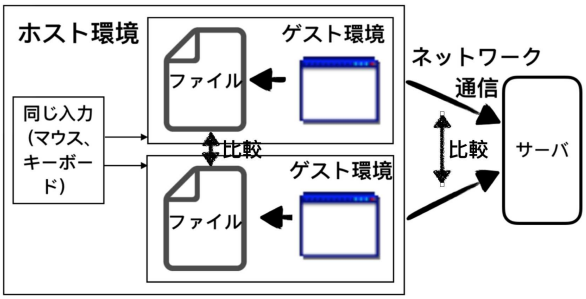

図 1: 双子の環境

や HTML5 の IndexedDB などのストレージを使う方法もある。本研究では双子のブラウザを用いてブラウザが作成する全てのファイルの差分およびブラウザが発信するネットワーク通信の内容の差分を調査する。その差分にユーザトラッキングのための情報が含まれる可能性が高い。その差分を削除、または修正することでユーザトラッキングを阻止することができると思われる。

3 コンテナによる双子の環境の実装

本研究では、環境内のファイルを調査するが、一般的な仮想マシンではホストはゲスト OS のファイルシステムを直接的アクセスできないという問題がある。そこで本研究では、コンテナという OS 層の仮想マシンを使う。

コンテナは、仮想化技術の一種である。VMware や Xen、KVM などの一般的な仮想マシンとは違い、コンテナはハードウェア層ではなく、OS 層の仮想マシンである。コンテナは Linux カーネルの cgroups と namespaces 機能を利用し、リソース管理と隔離を提供している。

本研究ではコンテナを実装する仕組みとして Docker を使う。 Docker では、 Overlay File System というファイルシステムが利用可能である。このファイルシステムではゲスト OS のファイルをホスト OS からアクセスできるため、ファイルの差分を取得することが容易である。

ゲスト OS が生成したファイルは Overlay File System の Upper Layer に保存される。したがって、Upper directory のみ読み込むことでファイル内容の変化を得ることができる。

4 ファイルの差分検出

図 2 に、ファイルの差分検出の仕組みを示す。まず、同じイメージを利用する 2 つのコンテナを起動し、対象となるプログラム (主に双子のブラウザ) を実行する、ホストからの入力を 2 つ複製して、2 つのコンテナを操作する。それからコンテナが生成したファイルをホスト OS で取得する。そして、ファイルの種類ごとにファイル内容に前処理を行い、その結果を diff コマンドに与える。

4.1 前処理とテキスト化

プログラムでよく利用される保存形式としてはテキスト形式、データベース形式、および、マーシャリング形式の 3 つがある。マーシャリング形式としてはよく JSON と XML がよく使われる。本研究では Web ブラウザ Firefox と Google Chrome のファイル保存形式を調査した。その結果、テキスト、JavaScript、JSON、XML、SQLite、BerkeleyDB、LevelDB の 7 種類あることがわかった。

コンテナが出力するファイルがテキストファイルであれば、そのまま diff コマンドで差分を取ることができる。しかしながら、JSON や XML では、そのまま diff コマンドに与えても大量の出力がなされ、目的とする個人情報が埋もれてしまうという問題がある。また、diff コマンドは、バイナリファイルを扱うことができない。

そこで本研究ではプログラムが生成したファイルに対して前処理を行い、diff で差分を取る前に識別しやすい形に変換する (図 2)。形式分類モジュールで、ファイルを形式で分類する。関係データベースは dump ツールでテキストを生成し、自動増加の主キー列を消して、属性内容でソートする。JSON ファイルの標準化するために、json.tool を利用した。

4.2 タイムスタンプの扱い

ネットワーク通信を行うプログラムは、様々なタイムスタンプをファイルに保存する。単純にファイルの差分を得ると、タイムスタンプの差によるものが大量に生成され、重要な差分が埋もれてしまう。

本研究では、タイムスタンプを次のように分類して扱う。

・リモート： リモートの通信相手が指定したもの。例えば、HTTP の Last-Modified:ヘッダに由来するもの。

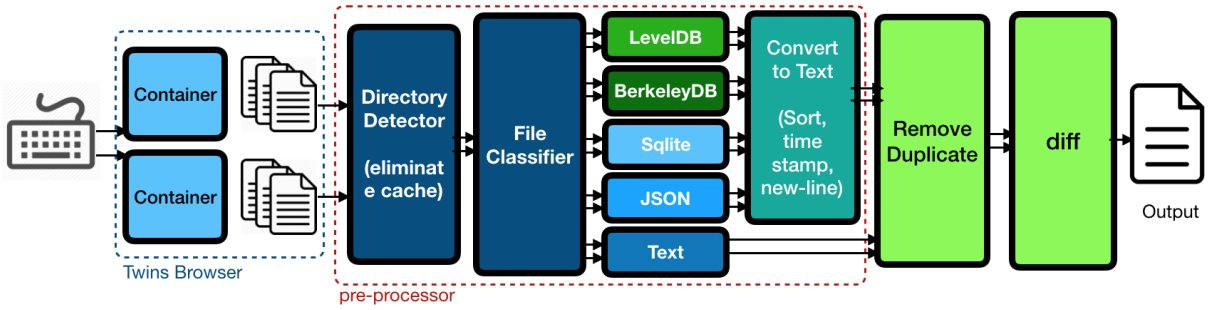

図 2: Docker Overlay Filesystem によるファイル差分検出

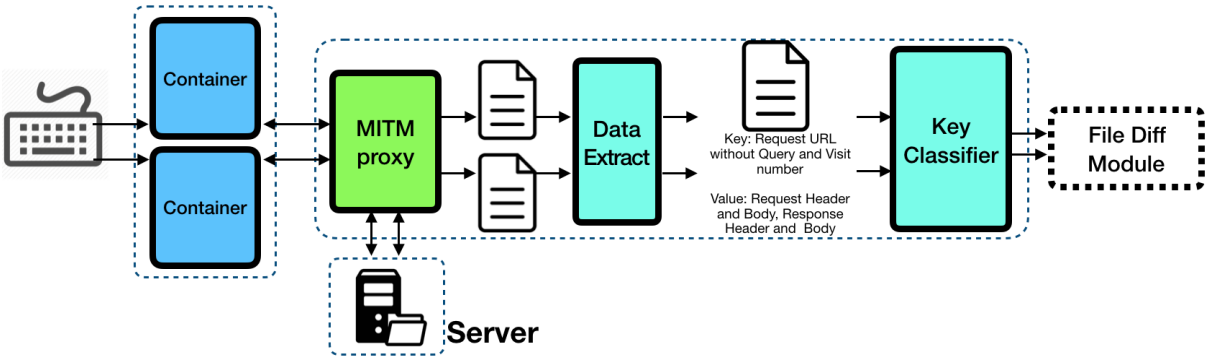

図 3: MITM-proxy を用いた HTTP のメッセージの差分検出

・ローカル: ローカルの OS からシステム・コールで取得したものに由来するもの。

リモート・タイムスタンプは、外部に発信されることが想定されている。例えば、HTTP の応答メッセージに含まれた Last-Modified. ヘッダの値は、同じコンテンツを再取得する時に、要求メッセージの if-Modified-Since. ヘッダに含まれて発信される。この値は、個人識別子としてユーザトラッキングに使われることがあることが知られている。

一方、ローカル・タイムスタンプは、外部に発信されなければ、個人識別子にはなり得ない。したがって、双子の環境の実装では、ローカル・タイムスタンプの違いを排除したい。本研究では、双子の環境で実行したタイム関連のシステム・コール gettimeofday() と clock_gettime() をオーバーライドする。そのため、本研究で作成した動的リンクライブラリを LD_PRELOAD で置き換える。置き換えたシステム・コールは、環境変数で指定された固定の日付を返す。

このような処理を行ったとしても、アプリケーションの内部で独自にシステム・コールで得たタイムスタンプを加工して利用していることがある。例えば、Firefox では、システム・コール clock_gettime() で得られたナノ秒単位の時刻に、1 から 4 まで加えた値を利用している。この問題を解決するために、本研究では、テキスト化した後、ローカル・タイムスタンプとそれを加工したものと思われる数字を定数で置き換える。

4.3 ランダム性の排除

プログラムは多く予想出来ない行動を行う。その結果、ファイルの内容に差が生まれ、本来検出したい個人情報を覆い隠してしまう。そこで、本研究では、そのようなランダムな行動を排除する。

まず、乱数によるプログラムのランダム行動を抑止するために、乱数デバイス/dev/urandom を置き換える。本研究では擬似乱数生成器デバイスドライバを作り、双子のコンテナのインスタンスに同じシードを与える。

マルチスレッドやマルチプロセスのアプリケーションのスケジュールも実行結果に影響を与える。Chrome ブラウザの Task モデルは、UI と IO スレッド以外にワーカスレッドが多数存在する。一つの作業がどのワーカスレッドにより実行されるか予想できない。その結果、Chrome ブラウザでの実験中には、TID のような予測できない結果が出力される。そして、PC のリアルタイムパフォーマンスによる動的にワーカスレッドの数を調整することもある。そこで本研究では、テキスト化してダンプする時に

スレッド識別子を含まないようにする。

5　ネットワークメッセージの差分

本研究では、まず、HTTPを対象としてネットワークメッセージをキャプチャする。コンテナ内で実行されるプログラムが送受信されているメッセージは、HTTPSにより暗号化されてることがある。そして、多数なHTML5やJavascriptの新機能もよく支持することが要求されている。そこで、本研究では、MITM-Proxy(Man-In-The-Middle-Proxy)[1]を使ってHTTPやHTTPSの通信をキャプチャする。MTIM-Proxyは HTTPS通信のキャプチャができ、Keep-Aliveなどの持続的接続機能もつける。

ネットワーク通信の差分取る仕組みを図3に示す。HTTP通信の内容を Key-Value の形式に変える。Keyとしては現在 Request の URL を Query String を排除した部分と訪問の回数を用いている。Value は残りの Query String, Header, body, Response Header, body にする。これらのデータを 4.1 節で述べたファイル差分検出モジュールに与える。

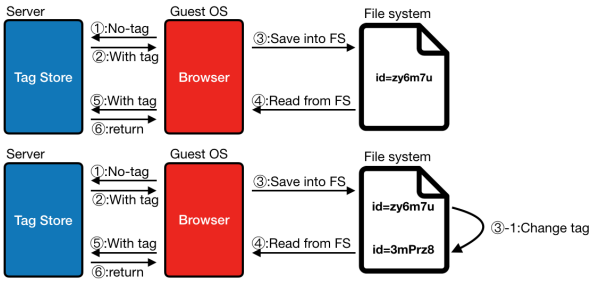

図 4: tag protection

6　差分内容の保護と隔離

トラッキングに使われる嫌疑があったタグを二回目サーバを訪問する時に、サーバが正しい認識できないような形に変える必要がある。図4のように、一回目の通信で生成した新しいタグをそして、タグの隔離効果をテストするため、修正前の元のタグはホストに保存しなければならない。ファイルとネットワーク通信の差分の対応は違う。

6.1　ファイルの差分の対応

ファイルの差分内容は全部 diff コマンドで見つかったから、出力ファイルは解析し、オリジナルファイルのふさわしい部分を処理する必要がある。ファイルの形によって処理手段は違うから、次のように扱う。

・テキストファイル:　テキストファイルの対応は、まず diff の出力ラインから差分タグを検出、差分タグを双子の環境の各ホスト環境のストレージに保存する。それからファイル名でゲスト OS のファイルシステムに対応したファイルを awk で差分タグを直す。直したタグは直した前のタグと同じように保存する。

・データベース:　テキストファイルと同じように差分タグを検出、保存する。タグの直すは awk ではなく、データベースの管理ツールを利用して簡単で直せる。関係型データベースは SQL の update を利用し、No-SQL は直接に Key-Value を直す。

・バイナリファイル:

6.2　ネットワーク通信の差分の対応

ネットワークの通信の差分タグは通信中にプロキシで直せるが、ブラウザの動作に影響があるから

6.3　ディレクトリのオプティマイズ

ネットワークの通信の差分タグは通信中にプロキシで直せるが、ブラウザの動作に影響があるから

7　実験

7.1　双子のブラウザが生成したファイルの差分

本研究室では Firefox と Google Chrome で双子のブラウザを実装している。今回、ユーザトラッキングを行っている代表的な Web サイトとして Google を選択し http://www.google.com/を訪問する実験を行った。これは、ログインを必要としない簡単なサイトである。

7.1.1　Firefox

まず Firefox ブラウザを双子のブラウザとして双子の環境で実行した。コンテナの全てのファイルの変化をリストアップしたところ、全部で 59 個のファイルが作成された。その中に 29 個が /.cache にある一時的なファイルのためファイル名によるフィルタ

[1] https://mitmproxy.org/

表 1: 差分がないファイルと差分があるファイルの数

	Firefox			Google Chrome		
	個人情報がない	差分がある		個人情報がない	差分がある	
		不明なバイナリ	テキストと扱えるバイナリ（行数)		不明なバイナリ	テキストと扱えるバイナリ（行数)
更新されたファイル	52	0	7(28)	98	5	47(114)
乱数生成器の置き換え	55	0	4(10)	128	5	17(46)
ローカルタイムスタンプの固定	55	0	4(8)	128	5	17(41)
前処理後	55	0	4(8)	136	5	9(15)

表 2: ファイルの差分の例

ファイル名	コンテナ 1	コンテナ 2
cookies.sqlite	value = 132=S1fjWe4xjUJJowuImYQyi...	value = 132=ablTT1GOqBlYlBX-MQ...
places.sqlite	guid = xhIxtPu6zAJ7	guid = IlfETnEs0Dr4
sessionstore.js	”docshellUUID”: ”{4c4508da-9468-430b-8eac-0484dcc43e5d}”	”docshellUUID”: ”{bfcff6ba-42c2-4731-a65c-ae1ba7b1cc0e}”
prefs.js	user_pref(”browser.slowStartup.averageTime”, 14971)	user_pref(”browser.slowStartup.averageTime”, 18103)

により自動的に除外された。残る 30 個のファイルに対して前処理を行って、テキスト化し diff コマンドで差分を取った結果は全部 28 行であった。4 章で述べた方法に従ってテキスト化した差分を削減した結果、最終的に差分は 8 行だった。識別できないバイナリファイルはなかった。

表 3 に、発見した差分の一部を示す。cookies.sqlite には、HTTP Cookies が保存されていることがわかる。places.sqlite は、訪問履歴を保持するファイルである。その中に、アクセスした URL が保存されている。このように、URL の中に、ユーザトラッキングに利用可能なタグが埋め込まれていることがわかる。

この実験の結果、提案手法により、ファイルを特定し、使用者の識別子を検出することができ、ユーザトラッキングに使われる個人識別情報が含まれることが確認された。

7.1.2 Chrome

2 番目の実験は Google Chrome で行った。結果を表 1 に表す。差分があるファイルは Firefox より多い。識別できないバイナリファイルが 5 個残された。

7.2 ネットワーク通信内容の差分

ブラウザが行うネットワーク通信の内容の差分を調査した。訪問したサイトは、7.1 節と同じである。

全部の通信の数は 21 個あり、テキスト形式のファイル以外に png が 6 個、ico が 1 個あった。20 個のメッセージの内容に差分があり、1 番目の通信だけは差分がなかった。ネットワーク通信の差分はタイムスタンプと重複を除き全部で 22 行あった。応答メッセージの内容に差分があるファイルが 2 個あった。それらの URL は www.google.com と www.google.com/gen_204 であった。また、URL が違う通信の数は 3 個あった。

メッセージの内容には cookies が現れている。これは、cookies.sqlite(表 3) にも含まれている。

8 関連研究

Qubes-OS[1] は高いセキュリティを実現するための OS である。Qubes-OS は仮想計算機モニタ (Xen) を用いて、アプリケーションを隔離された仮想実行環境で実行する。Qubes-OS には、複数の仮想実行環境のファイルを比較する機能はない。

Blink-Docker[2] はコンテナ中でブラウザを実行することで、Canvas Fingerprint を利用したユーザトラッキングからユーザを保護する。Canvas Fingerprint はハードウェアや OS のわずかな違いによってクラ

表 3: ネットワーク通信の差分の例

URL	コンテナ 1	コンテナ 2
www.google.com	Cookie:1P_JAR=2017-12-12-04,NID=119=BGA6eL4YT9Q...	Cookie:1P_JAR=2017-12-12-04,NID=rzmrga_L3s...
www.google.com/gen_204	ei:pdcsW5TSLIb28AXosLGYBA	ei:pdcsW-vCDMfS8QWq-I34Bg
ssl.gstatic.com/gb/images/i1_1967ca6a.png	age:253629	age:253628

イアントを特定する手法である。Blink-Dockerはコンテナ技術を利用し、毎回 Fingerprint を変えることができる。本研究では、通信内容を検査し、そのようなユーザトラッキングを検出したいと考えている。

9 まとめ

本研究では人間の双子の研究に参考した双子の環境を提案した。本研究では Docker コンテナを利用し、軽量で類似の仮想環境を作り、ファイルと通信内容に含まれる個人情報を検出する。

本研究室で開発された双子のブラウザを双子の環境で実行し、それらが生成するファイルの差分を調査するツールを実装した。また、Man-In-The-Middle Proxy を用いて、通信内容を比較するツールを実装した。現在の実装では扱えないバイナリファイルの形式がある。今後は、複雑なサイトや内容がよく変わるサイトも実験を行う。また、ネットワークとファイル間の関連も調査する。そして、ファイルや通信内容から不要な個人情報を自動的に削除するツールを実装する。

参考文献

[1] Qubes OS - A reasonably secure operating system: https://www.qubes-os.org/, accessed: 2017-11-22.

[2] P Laperdrix, W Rudametkin, B Baudry: "Blink: A moving-target approach to fingerprint diversification" 37th IEEE Symposium on Security and Privacy, Poster Session, [Online] http://www.ieee-security.org/TC/SP2016/poster-abstracts/59-poster_abstract.pdf, (2016).

[3] 張 世申, 新城 靖, 三村 賢次郎: "個人情報を含むファイルと通信を検出ための双子の環境の提案", 情報処理学会第 29 回コンピュータシステムシンポジウム ポスターセッション, 2 ページ (2017).

[4] 三村 賢次郎, 新城 靖, 張 世申: "Web サービスごとに隔離されたブラウジング環境の提案", 同上.