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[1]　三村　賢次郎[1]　新城　靖[1]

概要: インター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，意図せず個人情報や個人を識別できる情報をサービス提供者に送信することがある．本研究では，ユーザのプライバシの保護を目的として，ファイルやネットワーク通信からそのようなユーザトラッキングに使われる情報を検出する．そのため本研究ではコンテナ技術を利用し，双子の環境という仮想実行環境を提案する．双子の環境を利用する双子のブラウザを実装する．

キーワード: ユーザトラッキング,HTTP, コンテナ, ブラウザ

Zhang Shishen[1]　Kenjiro Mimura[1]　Yasushi Shinjo[1]

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．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．

[1] 筑波大学
University of Tsukuba

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 [4]．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

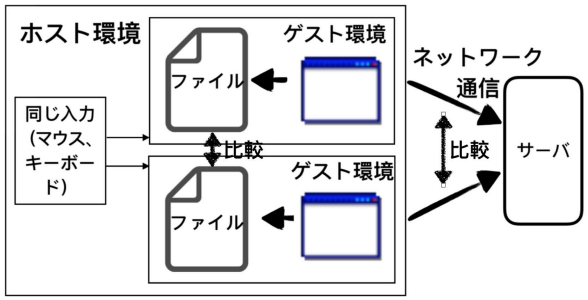

図 1　双子の環境

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．

2.1　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．

2.2　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．

2.2.1　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．

2.3　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．
- 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

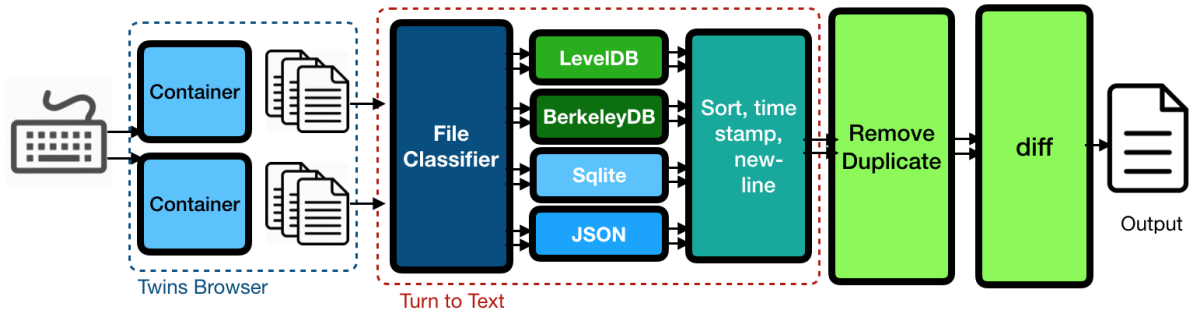

図 2　Docker Overlay Filesystem によるファイル差分検出

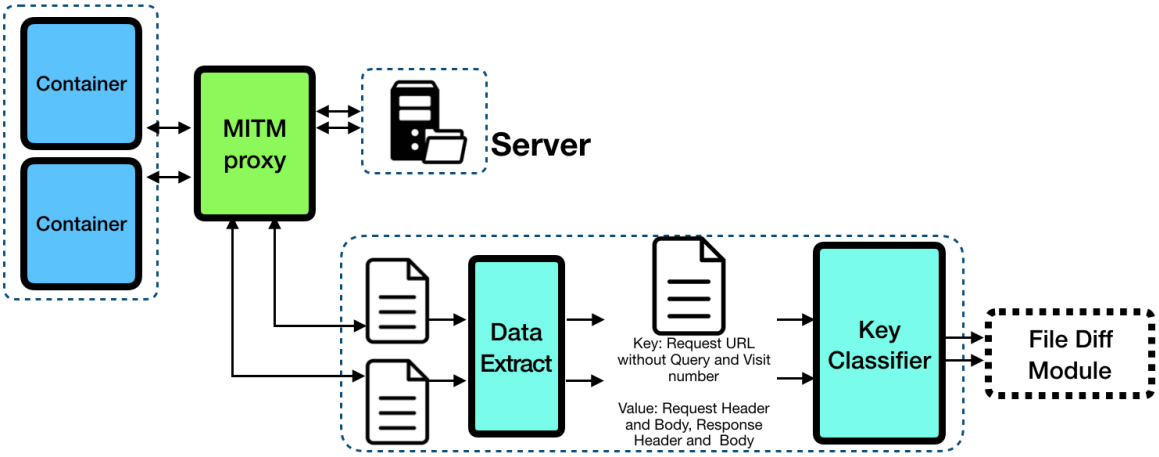

図 3　MITM-proxy を用いた HTTP のメッセージの差分検出

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．

2.4　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 [3]．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．

2.5　ネットワークメッセージの差分

コンテナが生成したファイルだけではなく，ネットワーク通信の内容に個人識別情報も多く存在する．これらのメッセージも差分を取って個人情報を検出したい．本研究ではブラウザアプリケーションを使って，ブラウザ通信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．以前の Web ブラウザと HTML のバージョンは，HTTP Request-Response に基づくだけが，多数のシナリオには効率が低いから，新しい HTML5 には多くの新しいプロトコルを提出した．また，既存の Request-Response の形の HTTP プロトコル (HTTP1.1) もいろんな制限があるし，今は HTTP2.0 が提出され，多くの場合で使用される．それ以外に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

[*1] https://mitmproxy.org/

プチャする．

2.5.1 Man-In-The-Middle Proxy によるネットワーク通信のキャプチャ

MTIM-Proxy は, 対話型の HTTPS プロキシであり，Python API を利用する簡単で HTTP と HTTPS の通信をキャプチャ，修正することができる．一般的な通信プロキシの機能の上で，いくつの HTML5 機能も加え，Keep-Alive などの持続的接続機能もつける．MITM-Proxy は以下の通信プロキシをよく支持する．

- Raw 通信 MITM-Proxy には TCP 通信のキャプチャとフィルタを実装した．
- HTTP と HTTPS の通信 HTTPS の通信のキャプチャは TLS 層を全部 transparent-proxy の形式でフィルタし，ユーザは HTTP 通信だけ見える．
- websocket の通信 HTML5 の新しいプロトコルとしてよく使われる．HTTP メッセージとは違い，双方向の通信である．MITM-Proxy には websocket 通信を支持し，Connection と通信のキャプチャができる．

他の通信型もいくつある．例えばメディア通信でよく使われた WebRTC(Web Real-Time Communications) プロトコル，Flash や Silverlight などのプラグインによる通信．これらの通信のキャプチャはまだ実装してない．

2.5.2 通信データのテキスト化と Dump

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．Dump した内容は HTTP 通信のヘッダ Content-Type や Content-Encoding による分類して，ふさわしい処理を行う．例えば”Content-Encoding: gzip”ヘッダの通信は圧縮した形を展開し，”Transfer-Encoding: chunked”ヘッダの通信は chunk の属性のよって，多数の Chunk を組み合わせる．それから，全部のテキストファイル，例えば html や javascript，css などのファイルはテキストファイルとして 2.2.1 節で述べたファイル差分検出モジュールに与える．テキストファイルではない通信，例えば png,favicon などの通信はバイナリファイルとして扱う．Websocket の通信は Full Duplex であるから，一つの URL が多数の通信を行う．Websocket の Key は URL と通信の回数，送受信方を用い，Value は通信の内容にする．

3. 双子のブラウザの実装

本研究では，個人情報と個人識別し以外の差分を最低限にするために 2 つのブラウザを同時に動かす．この双子のブラウザは，図 4 のように親ブラウザがあり，このブラウザを実際に操作し同じ操作を 2 つの子ブラウザに与える．

双子のブラウザは，大きく親ブラウザの操作を読み取る部分と子ブラウザを操作する部分に分けられる．ブラウザの操作を読み取る部分は，WebExtensions[*2] を用いてブラウザの機能を拡張することで実現する．この WebExtensions は，Google Chrome と FireFox でほぼ同じように動作する．発生した操作を検知するために，親ブラウザが表示する Web ページの DOM に対して予めイベントを登録しておく．イベントが発生した場合には，発生したイベント名 (クリックや入力など) と対象の DOM を送信するように登録しておく．対象の DOM の記述には，Xpath を用いる．また，イベントがどのタブで発生したか判別できるようにタブ ID も同時に送信する．

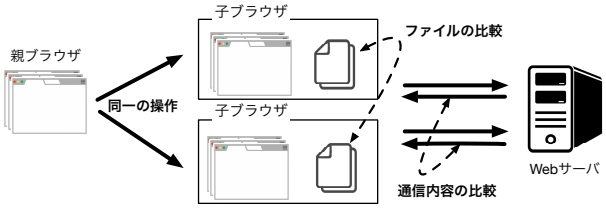

図 4 双子のブラウザ

ブラウザを操作する部分は，Selenium[*3] を用いて実装する．Selenium は，Web ブラウザの自動化用ツールであり，Java や Python, Node.js から Web ブラウザのマウスやキーボードを使った操作を入力する．子ブラウザは，親ブラウザで発生したイベントの種類に応じて，Selenium ライブラリを介して操作する．子ブラウザには，初めウィンドウが存在しないため Selnium によってタブを作成し，要求された URL を表示する．この時，タブのハンドラと送られてきた親ブラウザのタブ ID の対応を保持しておく．すでに親ブラウザで開かれたタブは，子ブラウザの対応するタブに操作をすることで，親と子ブラウザで複数のタブを協調する．

双子のブラウザの実装を図 5 に示す．この環境で同じ動作を行えるものを以下に示す．

- クリック
- フォームの input 要素の入力
- スクロール
- ウィンドウのリサイズ

4. 個人識別情報の削除

2 章で述べた方法で個人識別情報を含むファイルとネットワーク通信を検出した後，本研究では，それを削除するツールを実装する．

4.1 ファイルから個人識別情報の削除

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，多くのファイルをアクセスする．個人識別情報は，

[*2] WebExtensions: https://developer.mozilla.org/ja/Add-ons/WebExtensions
[*3] Selenium: http://www.seleniumhq.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)

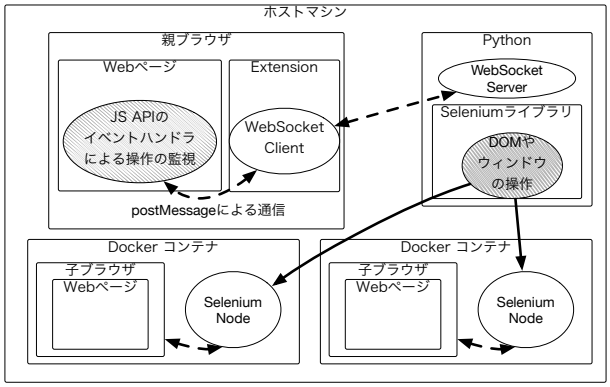

図 5　Selenium による双子のブラウザの実装

これらのファイルのうち，更新されるものに保存される．本研究では，**??**節の手法により，更新されるファイルを特定することができる．これらの更新されるファイルを削除すれば，個人識別情報も削除されるが利便性が大きく低下することがある．たとえば，Web ブラウザでは，ブックマークや個人の設定を保存するファイルが更新されるが，それらを全て削除すると，利便性が低下する．

そこで本研究では，更新されるファイルを，次の3 種類に分類する．

- アプリケーション自身が個人識別情報を削除する機能を提供しているもの．たとえば，Web ブラウザなら，cookie を終了時に削除する機能がある．
- ファイル全体を削除すると問題が生じるもの．
- ファイル全体を削除しても問題が生じないもの．

本研究では，ファイル全体を削除すると問題が生じるものについて，内部の個人識別情報を特定し，削除するか，ランダムな値で置き換えるツールを作成する．

本研究では，Web ブラウザの次のファイルから個人識別情報を削除するツールを作成する．

- Firefox cookies.sqlite persistant cookie の value をランダムな値で置き換える．
- Firefox pref.js 時刻に関する行 (browser.slowStartup.averageTime 等) を削除する．
- Firefox places.sqlite guid をランダムな値で置き換える．
- Firefox sessionstore.js docshellUUID をランダムな値で置き換える．principalToInherit_base64 をランダムな値で置き換える．

本研究では，Web ブラウザの次のファイルを削除とタグの書き換えをしても問題が生じないことを確認した．

- Firefox cookies.sqlite ファイルを直接に削除と persistant cookie の value をランダムな値で置き換える両方とも問題がない．ただし，value を書き換えてもう一度ブラウザをオープンしサイトを訪問すると，値が更新された．
- Firefox pref.js ファイルを削除するとブラウザは普通に作動するが，console から WARNING がいくつ出る時刻に関する行 (browser.slowStartup.averageTime 等) を削除すると問題がない
- Firefox places.sqlite ファイルを削除と guid をランダムな値で置き換えるには問題がない．ただし，guid をランダムな値で置き換えて，もう一度ブラウザをオープンしサイトを訪問すると，値が更新されてない．

- Firefox sessionstore.js docshellUUID をランダムな値で置き換えると principalToInherit_base64 をランダムな値で置き換えるには問題がない

4.2 ネットワーク通信から個人識別情報の削除

ファイルからの個人識別情報の削除は，プログラムを一度終了しなければ行うことができない．たとえば，Web ブラウザ Firefox の places.sqlite に含まれる guid は，本研究のツールで削除できる．しかし，ブラウザの実行中には，このツールは動作しない．

アプリケーション自身が，個人識別情報の送信を抑止する機能を持っていることがある．たとえば，Web ブラウザは，cookie を送信しない機能や，検索結果の URL から個人識別情報を削除する拡張機能を持っている．Web ブラウザについては，このような既存の機能を利用すれば，かなりの個人識別情報を削除することができる．しかしながら，Web ブラウザの機能には，ユーザがアクセスするページではないネットワーク通信を扱えないという限界がある．たとえば，Web ブラウザが SSL/TLS の証明書を検証するために Online Certificate Status Protocol(OCSP) に基づき行う通信を扱えない．また，Web ブラウザ以外のアプリケーションについては，ネットワーク通信から個人識別情報を削除する機能は開発されていない．本研究では，MITM-Proxy を用いてネットワーク通信から個人識別情報を削除する．具体的には，次のようなメッセージから個人識別情報の削除する．

- ファイルシステムに検出した個人識別情報
  個人情報はネットワークアプリケーションにより，ファイルに保存したデータを悪意のあるサーバに送信することがある．この状況を防ぐため，ファイルで検出した個人識別情報はネットワーク通信の内容と比較して，同じである部分をネットワーク通信内容から削除する必要がある．
- サーバから送信した情報
  cookie などの個人情報は，まず HTTP 通信よりサーバからクライアントに送信する．この原因でサーバからのメッセージを通信中に検出し，ファイルと同じように diff で差分を取った部分は，個人識別情報である可能性が高い．これらの差分情報を記録して，クライアントが送信したメッセージをフィルタする．
  たとえば HTTP Response のヘッダには “Set-cookie:”行があった時に，クライアント端に個人識別子になる cookie が保存される．しかし全部の cookie がユーザトラッキングに使われるではなく，ログインやショッピング用に使われた session のストアと，ユーザのセッティングに使われる場合もある．これらの cookie を削除するとブラウザの使用には問題が生じるかもしれない．そこで本研究は，サーバの response についた cookie の expiry を検出，長期間保存した cookie を記録し，クライアントの response の中に記録した cookie が含まれた時に HTTP ヘッダの cookie ラインを削除する．

個人情報の検出と通信中の内容削除は，HTTP Request と HTTP Response の形で実行する．HTTP 通信の中に，もっとも重要な部分は HTTP ヘッダである．ヘッダには多くの通信のメタデータが含む．

Web サーバは，IP アドレスやブラウザのフィンガープリント 等のメタ情報を使ってユーザトラッキングを行うこともある．送信するメッセージの内容だけを操作しても，それには限界がある．本研究では，IP アドレルを隠すことについては，Tor 等の既存の手法と組み合わせて用いることにする．

5. 実験

5.1 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．

5.1.1 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．2.2 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．

5.1.2 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．

表 3　ネットワーク通信の差分の数

全部の通信数	21
差分がある通信	20
応答に差分がある数	2
タイムスタンプと重複を排除の差分	22(行)

表 4　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

5.2　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，5.1 節と同じである．結果は**??**に示す．

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．

6.　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．

7.　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．実装した双子のブラウザを利用し，個人情報と個人識別子である可能性が高いユニークなタグがいくつ検出した．そして，個人識別子を変更の方で保護する．

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] Javier Verd, Juan Jos Costa, Alex Pajuelo: ”Dynamic Web worker pool management for highly parallel javascript web applications” Concurrency and Computation: Practice and Experience, 28(13):35253539, September 2015.
[4] 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).
[5] 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.
[6] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, Claudia Diaz, ”The Web Never Forgets: Persistent Tracking Mechanisms in the Wild”, CCS ’14 Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security Pages 674-689.