HTTP プロキシ を利用した Cookie Bomb 攻撃の検知手法の提案

2019SC022　河合航希　　2019SC027　倉橋良汰
指導教員：石原 靖哲

1　はじめに

現在，Web サイトでは Cookie を利用することが多い．Cookie とは Web サイトにアクセスしたユーザの情報を，ブラウザに一時的に保存するための仕組みである．Cookie を利用することで，パスワードや ID の入力回数を減らすことができ，Web サイトの利便性の向上につながる．

一方，Cookie がセキュリティ上の脅威につながることがあり，その一つとして，Cookie Bomb 攻撃 [1] が挙げられる．Cookie Bomb 攻撃とはブラウザに巨大なサイズの Cookie を設定させる攻撃である．それ以降，ブラウザが Web ページにアクセスするたびに，巨大な Cookie を送信することになるため，Web ページが利用できなくなる．Cookie Bomb 攻撃はユーザが Cookie Bomb 攻撃に気づき，Cookie を削除することで対処できる．しかし，ブラウザに表示されるエラーメッセージだけでは Cookie Bomb 攻撃を受けていると気づきにくく，適切に対処することが難しいという問題点がある．

本研究では HTTP プロキシを介した Cookie Bomb 攻撃の対策と，その効果について検討する．Cookie Bomb 攻撃を検知する手法の研究として，岡澤ら [2] の研究がある．その研究ではブラウザの拡張機能を利用した Cookie Bomb 攻撃の検知手法を提案している．それに対し，本研究ではブラウザ拡張機能ではなく，プロキシを用いた検知手法を提案する．複数の HTTP レスポンスにおける Set-Cookie ヘッダ内のバイト数を数え，Server ヘッダから得たサーバ情報をもとに Cookie のデータサイズの上限値を設定し，その上限値よりも Cookie のバイト数が上回っていた場合，エラーの画面を出し通信を遮断するという手法である．ブラウザ側で検知する手法に対して，プロキシ上で検知する手法にすることで，サーバごとに異なる Cookie のデータサイズの上限値に対応できるようになるため，Cookie Bomb 攻撃をより正確に検知する事が出来る．

結果として，本研究で提案したプロキシを介した通信を行うことで，サーバごとに対応したヘッダに含めることのできるデータサイズの上限値を設定し，Cookie Bomb 攻撃を検知する事ができた．また，クロスサイトスクリプティングなど他の脆弱性を利用した攻撃も検知する事ができた．

2　関連研究

岡澤ら [2] は Cookie Bomb 攻撃を検知するための手法として，Chrome の拡張機能を用いたものを提案した．

まず，ブラウザが HTTP リクエストを送信しようとしたときに，Cookie ヘッダの有無を確認し，Cookie のサイズの計算を行う．その値が基準値となる 8000bytes を超えた場合ブラウザに警告メッセージとユーザがアクセスしようとしているサイトの URL，Cookie ヘッダのサイズをブラウザの画面に表示する．その後検知した巨大な Cookie を削除し，ブラウザからの HTTP リクエストを遮断する．

課題として，Cookie サイズの上限値が 8000bytes 固定であるため，サーバの種類によっては Cookie Bomb 攻撃を誤って検知してしまう可能性があるとしている．また，Chrome の拡張機能で作成したプログラムであるため，今後の展望として Firefox や Microsoft Edge 用の拡張機能を開発していく予定としている．

また，Watanabe ら [3] の研究では Cookie Bomb 攻撃は AppCache を用いた MITM 攻撃などにも利用できるとしている．Cookie Bomb 攻撃は AppCache の Web ページを書き換える機能と組み合わせることでより高度な攻撃を確立させることができる．

3　Cookie Bomb 攻撃

3.1　Cookie

Cookie が設定される仕組みについて図 1 に示す．ウェブサーバへのアクセスが 1 回目の時には，ブラウザに Cookie は設定されていない．ウェブサーバから Cookie が送信されることで，Cookie がブラウザに保存される．次にウェブサーバにアクセスする際に，保存された Cookie が送信されることで，同じ利用者のブラウザであると判断できる．2 回目の通信以降は Cookie により，ログイン状態が一定期間保持されるため，何度もログイン情報を入力しなくても Web ページを利用することができる．

さらに，Cookie を設定する際には属性を追加することができる．例えば，Domain 属性を追加すると，指定されたドメインのサブドメインにも Cookie が保存される．Expires 属性を追加すると，その Cookie の有効期限を指定することができる．

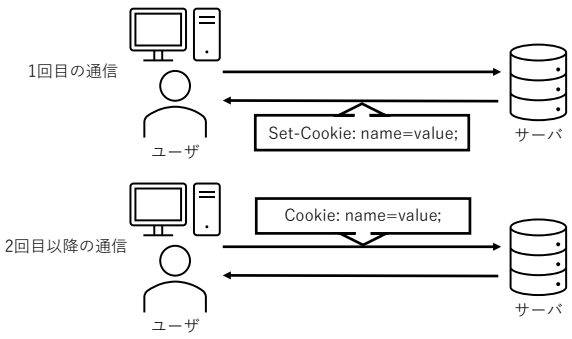

図 1　Cookie の設定

3.2 Cookie Bomb 攻撃の手法

Homakov[1] は Cookie サイズの上限値を利用した攻撃を発見し，Cookie Bomb 攻撃と名付けた．Cookie Bomb 攻撃とは，攻撃者が被害者のブラウザに巨大なサイズの Cookie を設定させる攻撃である．それ以降，ブラウザが Web ページにアクセスするたびに，巨大な Cookie を送信することになるため，Web ページが利用できなくなる．

Cookie Bomb 攻撃の手法を図 2 に示す．1 回目の通信で巨大な Cookie をたくさん生成し，サーバソフトウェアの HTTP リクエストヘッダに含めることのできるデータサイズの上限値より大きくする．2 回目以降の通信ではブラウザは保存されているすべての Cookie の情報を HTTP リクエストの Cookie ヘッダによって送信するため，サーバソフトウェアのデータサイズの上限値を超えてしまう．HTTP リクエストヘッダが長すぎる場合，サーバは応答しなくなるため，Web ページを利用できなくなる．このサーバソフトウェアのデータサイズの上限値はサーバソフトウェアごとに異なり，デフォルトの場合，Apache では 8192bytes，Nginx では 8191bytes となっている．また，Cookie 情報の読み込みや書き換えを行う必要があるため，クロスサイトスクリプティングや HTTP ヘッダインジェクションとともに Cookie Bomb 攻撃が行われることが多い．

クロスサイトスクリプティングとは，脆弱性がある Web サイトを用いて，攻撃者がその Web サイトへ誘導するための URL に悪意のあるスクリプトなどを仕掛けることで，その Web サイトにアクセスしたユーザを攻撃することである．この攻撃を利用して Cookie Bomb 攻撃が仕掛けられる場合，攻撃者がクロスサイトスクリプティングに対して脆弱性のある Web サイトを発見し，その Web サイトの URL に巨大な Cookie を保存させるスクリプトを仕掛ける．その URL を通じてユーザが Web サイトにアクセスすると，巨大な Cookie を保存させるスクリプトが実行され，ユーザは Cookie Bomb 攻撃の被害に遭ってしまう．

HTTP ヘッダインジェクションとは，Web アプリケーションの脆弱性を用いて，HTTP リクエストに改行コードである %0d%0a を含む不正な文字列を挿入することで，Web サーバから送られる HTTP レスポンスを改ざんする攻撃である．HTTP ヘッダインジェクションで Cookie Bomb 攻撃を仕掛ける場合，攻撃者は HTTP ヘッダインジェクションに対して脆弱性のある Web サイトを発見し，その URL のクエリパラメータに改行コードである %0d%0a を含めることで，Set-Cookie レスポンスヘッダが新たに生成されるようにする．その URL を通じてユーザが Web サイトにアクセスすると，Set-Cookie ヘッダによって巨大な Cookie が保存され，ユーザは Cookie Bomb 攻撃の被害に遭ってしまう．

Cookie Bomb 攻撃はユーザが Cookie Bomb 攻撃に気づき，Cookie を削除することで対処できる．しかし，被害者のブラウザに表示されるエラーメッセージだけでは Cookie Bomb 攻撃を受けていると気づきにくく，適切に対処することが難しいという問題点がある．

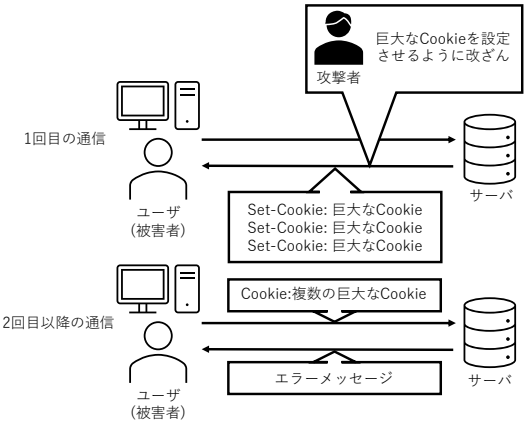

図 2 Cookie Bomb 攻撃

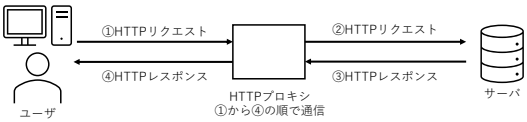

図 3 HTTP プロキシを介した通信

4 HTTP プロキシ

HTTP プロキシとは，ユーザからサーバへのアクセス要求を受信すると，自らが接続元となってユーザの代理として接続を行うサーバのことである．HTTP プロキシを使用した際の通信は，図 3 に示したような，ユーザとサーバ間での通信にプロキシを介した形となる．このような通信を行うことで，通信を監視して危険なサイトへのアクセスを防ぎ，接続先からはプロキシ自身が通信しているように見えるため匿名性が高まるなどの利点がある．

本研究では HTTP プロキシとして，個人及び少人数での使用を前提として作成されたものである Tinyproxy を使用する．他にも，HTTP プロキシとして Apache や Squid の利用も検討した．その上で，プロキシでありながらソースは比較的簡単であるためユーザごとにカスタマイズしやすいという点，システムリソースをほとんど使用せず，個人での使用に向いているという点から Tinyproxy を使用する．

5 提案する Cookie Bomb 攻撃の検知手法

5.1 本研究と先行研究の特徴と比較

本研究では HTTP プロキシを利用して Cookie Bomb 攻撃を検知する手法を提案する．そこで，先行研究のブラウザ拡張機能を利用した検知手法と本研究の比較を行う．

本研究で提案する手法は HTTP プロキシを使用することで HTTP リクエストと HTTP レスポンスを監視し，攻撃を検知する手法をとっている．長所はブラウザの種類を

問わず攻撃を検知する事ができること，プロキシならではの柔軟な対応が可能であることが挙げられる．また，事前にサーバ情報を取得することで，サーバごとに対応した上限値を設定できる点も長所であるといえる．短所は設定や運用についてはやや負担が大きくなることが挙げられる．

岡澤らの研究で提案された手法は Google Chrome の拡張機能を使用することで HTTP リクエストを監視し，攻撃を検知する手法をとっている．長所はブラウザ拡張機能であるため，利用が簡単であることが挙げられる．短所は Google Chrome 以外のブラウザでは利用できないこと，Cookie Bomb 攻撃を受けていると判定する値が 8000bytes 固定であるため誤検知の可能性が高いことが挙げられる．

拡張機能を使用し，なおかつサーバ情報を取得することで Cookie の上限値がサーバごとに対応したものを設定できるようになった場合，個人で使用する際には拡張機能を利用した検知手法の方が利用しやすいといえる．しかし，拡張機能は組織や多人数で使用する場合には個々人ごとに設定が必要となるため，プロキシを利用する方が手間が少なくなり，運用コストも抑えられる．また，組織で利用する場合，プロキシ側の設定で一括で管理できることはプロキシならではの利点であるといえる．

5.2 攻撃を検知するために必要なこと

Cookie 情報は HTTP レスポンスの Set-Cookie ヘッダによって送信され, ブラウザに保存される．そして, HTTP リクエストの Cookie ヘッダによってブラウザに保存された Cookie が送信される．また，通信先のサーバ情報が含まれているのは HTTP レスポンスである．接続先のサーバの種類によって Cookie Bomb 攻撃を誤って検知してしまう可能性があるという関連研究の課題があった．その課題に対して，この Server ヘッダから使用しているサーバソフトウェアの情報を取り出し，そのサーバソフトウェアの HTTP リクエストヘッダに含めることのできるデータサイズの上限値と Cookie のデータサイズを比較して Cookie Bomb 攻撃を受けているか判定できるようにする必要がある．

また，Cookie Bomb 攻撃は，通信の際に巨大な Cookie の情報を送信する事でエラーとなることを利用した攻撃であるため，図 1 のように 1 回目の通信ではエラーは起きず，2 回目以降の通信を行った際にエラーとなる．この 1 回目の通信から 2 回目の通信までに Tinyproxy を介して情報を取得し，Cookie に対して適切な対処をする必要がある．

5.3 検知する手法の説明

HTTP レスポンスから Server ヘッダと Set-Cookie ヘッダの情報を取得し，Cookie Bomb 攻撃を受けているか判定する手法について説明する．

まず，ブラウザに保存されている Cookie 情報を HTTP リクエストの Cookie ヘッダから取得する．取得した Cookie ヘッダの bytes 数を cookie.txt ファイルに書き込み，値を保存しておく．

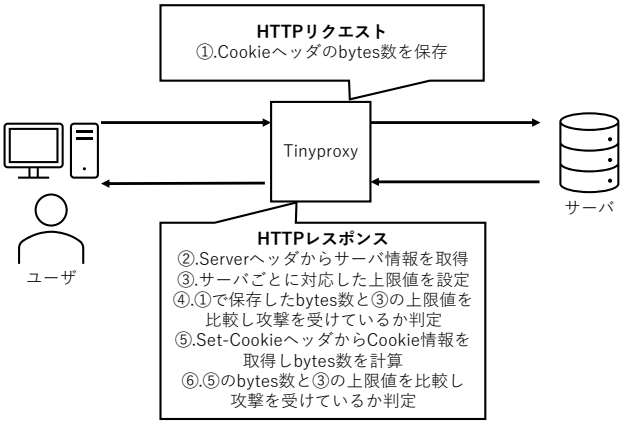

図 4 検知手法の一連の流れ

HTTP レスポンスにはサーバ情報を保存する Server ヘッダと Cookie 情報を保存する Set-Cookie ヘッダがある．Server ヘッダからサーバ情報を取得し，そのサーバごとに対応したヘッダに含めることのできるデータサイズの上限値を設定する．Apache の場合は 8192bytes，Nginx の場合は 8191bytes，LiteSpeed の場合は 32768bytes を設定する．この時，HTTP リクエストの Cookie ヘッダの bytes 数が保存されている cookie.txt を呼び出し，その値がすでにサーバのヘッダに含めることができるデータサイズの上限値を超えていた場合，エラーの画面とエラーメッセージを表示し，接続を遮断する．

次に，Set-Cookie ヘッダから Cookie 情報を取り出し，Cookie の bytes 数を計算する．Cookie Bomb 攻撃は HTTP リクエストの際に送信される Cookie ヘッダの上限値を用いた攻撃である．そのため HTTP レスポンスから HTTP リクエストの際に送信される Cookie ヘッダを計算する必要がある．Set-Cookie ヘッダにドメインや有効期限などの属性が含まれていた場合，セミコロン以降を削除し，bytes 数を計算したものから「Set-Cookie」分の 10bytes を引いたものをその Cookie の bytes 数とする．Set-Cookie ヘッダに属性が含まれていなかった場合，bytes 数を計算したものから「Set-Cookie」と改行コードを含めた 12bytes を引いたものをその Cookie の bytes 数とする．最後に，複数の Cookie のバイト数の合計とサーバのヘッダに含めることができるデータサイズの上限値を比較し，接続の判定を行う．Cookie のバイト数がデータサイズの上限を超えていた場合，エラーの画面とエラーメッセージを表示し，接続を遮断する．処理の一連の流れは図 4 のようになっている．

この方法は複数の Cookie を使用することができ，なおかつ Cookie Bomb 攻撃を検出することができる．サーバごとに対応した Cookie のデータサイズの上限値によって判定するため，より正確に Cookie Bomb 攻撃を受けてい

るかどうかを判定することができる．さらに，HTTP レスポンスで攻撃が検知された場合は，ブラウザに Cookie が保存される前のサーバからブラウザへの通信を遮断するため，巨大な Cookie がブラウザに保存されることはない．

6 動作確認

実際の攻撃に近い方法で攻撃を受けた場合，本研究で提案する手法で検知できるのか，動作確認を行う．

6.1 サーバごとの攻撃の対処についての検証

Cookie Bomb 攻撃をサーバごとに対応したヘッダに含めることのできるデータサイズの上限値を設定し，攻撃を検知する事ができるか検証する．8192bytes の Cookie を設定する Web サイトと 8191bytes の Cookie を設定する Web サイトを作成し，これらの Web サイトにプロキシを通じてアクセスする．これをサーバソフトウェアが Apache の場合と Nginx の場合で，プロキシが正確に攻撃を検知できるか検証を行う．Apache の場合は 8192bytes の Cookie を設定する Web サイトにアクセスするとプロキシがエラーの画面とエラーメッセージを表示し，接続を遮断した．Cookie が設定される前に接続を遮断するため，Cookie はブラウザに保存されなかった．8191bytes の場合はプロキシを介していない場合と同様に Cookie は保存されるが，次回以降のアクセスでも特に問題なくアクセスできた．Nginx の場合は 8192bytes と 8191bytes のどちらの Web サイトの場合でもプロキシがエラーの画面とエラーメッセージを表示し，接続を遮断した．こちらもブラウザに Cookie は保存されなかった．

6.2 他の脆弱性を利用した攻撃の対処についての検証

この検証のプログラム及び攻撃手法は [4] を一部参考にしている．

まず，クロスサイトスクリプティングを利用した Cookie Bomb 攻撃を検知できるか検証する．入力パラメータ中のデータが適切に JavaScript の文字列リテラルとしてエスケープされないように設計した脆弱性のある Web サイトを作成した．そのサイトの URL の URI の部分に HTML の script タグを利用し巨大な Cookie を設定するスクリプトを書き，その URL にアクセスするとスクリプトが実行されるよう細工した URL を用意する．この細工した URL にプロキシを通じてアクセスした場合，攻撃を検知できるか検証した．結果としては，Cookie Bomb 攻撃を検知する事ができた．Set-Cookie ヘッダを使用しないため Cookie は保存されてしまうが，ブラウザに保存されている Cookie がサーバごとに対応したヘッダに含めることのできるデータサイズの上限値を超えたため，エラーメッセージを表示し，接続を遮断した．

次に，HTTP ヘッダインジェクションを利用した Cookie Bomb 攻撃を検知できるか検証する．Perl で指定した Web サイトにリダイレクトする Web サイトを作成する．そして，そのサイトの URL の URI の部分に %0d%0a と改行コードを含ませ，Set-Cookie ヘッダを書き込む．そうすると，Set-Cookie ヘッダとして HTTP レスポンスヘッダに Cookie 情報が書き込まれるようになる．このようにして巨大な Cookie を保存するよう細工した URL にプロキシを通じてアクセスした場合，攻撃を検知できるか検証した．結果としては，Cookie Bomb 攻撃を検知する事ができた．今回の実験ではリダイレクト先の Web サイトで別の Cookie を保存することを想定しており，リダイレクト前のサイトとリダイレクト後のサイトで 2 回に分けて Cookie が送信される．この時，どちらかの Cookie が上限値を超えていないと Cookie Bomb 攻撃とは判定されず，ブラウザに Cookie が保存されてしまう．そのため，今回の検証は，ブラウザに保存されている Cookie がサーバごとに対応したヘッダに含めることのできるデータサイズの上限値を超えたため，エラーメッセージを表示し，接続を遮断するという挙動となった．

7 まとめと今後の展望

本研究では，Tinyproxy を用いて，ユーザが Cookie Bomb 攻撃を受けているかを判定し，対処するプロキシを構築した．検知手法として HTTP プロキシを構築し，HTTP レスポンスからサーバ情報と Cookie 情報を取得し攻撃を受けているか判定する方法を提案した．関連研究の課題であった一つのヘッダごとに保存できるデータサイズの上限値がサーバごとに異なるという課題に対し，事前にサーバ情報を取得することでそれらに合ったデータサイズを設定することが可能となった．そして，クロスサイトスクリプティングなどの脆弱性を利用した実際の攻撃に近い方法で Cookie Bomb 攻撃を検知できるか実験を行った．その結果，URL にスクリプトが含まれていた場合や巨大な Cookie が既にブラウザに保存されていた場合にも Cookie Bomb 攻撃を検知する事ができた．

今後の展望として，現状対応しているサーバが Apache, Nginx, LiteSpeed の 3 つであるため，さらに対応するサーバを増やすことが挙げられる．

参考文献

[1] Egor Homakov. Cookie bomb or let's break the internet. `https://homakov.blogspot.com/2014/01/cookie-bomb-or-lets-break-internet.html`.

[2] 岡澤佳寛, 齊藤泰一. Cookie Bomb 攻撃を検知するブラウザ拡張機能. 2022 年暗号と情報セキュリティシンポジウム，2C1-4, 2022.

[3] Takuya Watanabe, Eitaro Shioji, Mitsuaki Akiyama, and Tatsuya Mori. Melting pot of origins: Compromising the intermediary web services that rehost websites. In *Network and Distributed Systems Security (NDSS) Symposium 2020*, 2020.

[4] 徳丸浩. 体系的に学ぶ安全な Web アプリケーションの作り方　第 2 版. SB クリエイティブ株式会社, 2018.