urandom

vol.4

Computer Security Magazine

mitmproxy入門 *op*

公平なランサムウェアプロトコル *yyu*

目次

1

mitmproxy 入門

1　はじめに

1.1　この記事について

コンピューターセキュリティにおける脆弱性調査やソフトウェア開発の作業の中で、クライアントとサーバーの間の通信内容を確認したり改変する必要に迫られる事がしばしばあります。HTTP/HTTPS 通信における通信内容の確認・改変では、Burp Suite、Charles や Fiddler と言ったいわゆるローカルプロキシーツールを使い、通信を中継しつつ、中継点で内容を確認・改変する手法がよく取られます。このような中継点で通信に介入する手法は、Man In The Middle Attack（MITM）と呼ばれています。この記事では、MITM で通信内容を確認したり改変したりできるプロキシーの一つである mitmproxy について基本的な操作等を説明し、HTTPS 通信の MITM についても触れます。

1.2　Man In The Middle Attack とは

通信の文脈で Man In The Middle Atatck（MITM）とは、二者間の通信を第三者の攻撃者が傍受・改ざんするための攻撃手法を指す言葉です。日本語では中間者攻撃と呼ばれています。

図 1に示す図は、MITM を受けていない通常の通信の図です。通常の通信では、通信の当事者である二者の他には、通信の内容に関与する者はいません。対して図 2に示す図が、MITM を受けている状態の通信の図です。MITM では、通信当事者である二者間の通信に第三者の攻撃者が何らかの方法で割り込みます。攻撃者は一方が送信した通信文を一旦受け取った上で、それを相手方に中継します。この中継作業を双方向に行うことで、二者間での通信を成立させつつ、第三者の攻撃者が通信文をのぞ

3.2 mitmproxy

概要

ソフトウェアの名前そのままのコマンドです。CUI で対話的に操作しながら、HTTP/HTTPS 通信に対する MITM が行えます。以下では CUI の基本操作と一部機能の操作を説明します。説明の中で登場するショートカットキーは case sensitive です。

基本操作

オプション無しで mitmproxy を起動すると、図 3に示すようなブランクな画面が立ち上がります。右下に `[*:8080]`との表示がありますが、これは全てのネットワークインターフェースの 8080/tcp で、プロキシーとして待機している事を意味します。つまり、mitmproxy はこの段階で既にプロキシーとして動作しています。

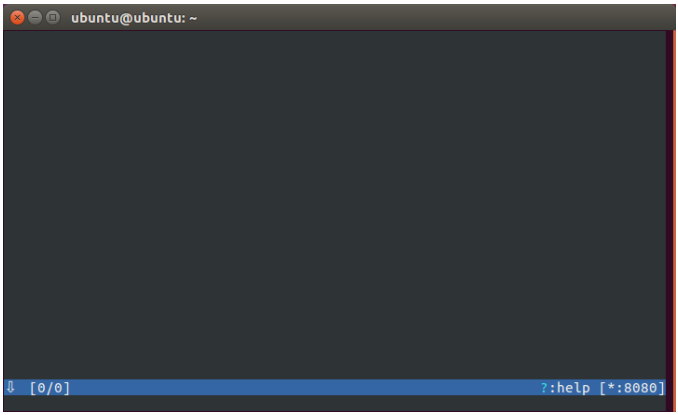

図 3: mitmproxy の画面（起動直後）

ここで、2.3の通りプロキシー接続先として mitmproxy を指定した Firefox で `http://www.example.com` を開いてみます。通信に問題が無ければ、Example Domain と見出しが付いたページが開くはず[*5]です。この時、mitmproxy の画面には図 4のように `http://www.example.com` に対する GET リクエスト・レスポンスの組と、その配下の favicon.ico に対する GET リクエスト・レスポンスの組が記録されているかと思います。このリクエストとレスポンスの組（通信一回分のトランザクション）を、

[*5] 2017 年 8 月 4 日時点

2

公平なランサムウェアプロトコル

1　はじめに

ランサムウェア（*Ransomware*）とは悪意のあるプログラムの 1 つです。感染すると被害者のコンピューターに保存されているデータを暗号化し、それを復号する対価として Bitcoin などのお金を要求するプログラムです。最近では “WannaCry” というランサムウェアが世界的に流行し有名となりました。ところが単純な疑問として、ランサムウェアの指示通りに Bitcoin などを送金したとして、はたして暗号化されたデータをきちんと復号してもらえるのでしょうか？ つまり従来のランサムウェアはお金もデータも失ってしまう可能性があるといえます。

本稿ではランサムウェアの作者であり Bitcoin を得たいアリスと、アリスのランサムウェアに感染してデータを暗号化されてしまったボブがいるとします。ボブはアリスへ Bitcoin を支払ってでもデータを復号したいですが、この 2 人の間には次のようにな不正が考えられます。まずアリスはボブから Bitcoin を受け取ったにも関わらず、暗号文を復号するための鍵を渡さない可能性があり、またボブはアリスが暗号文を復号するための鍵を渡したにも関わらず、Bitcoin を送金しない可能性があります。

一般的に、ランサムウェアの交渉はランサムウェアの作者であるアリスの方が優位であるため、アリスの不正が発生しがちです。本稿ではこのようなアリスとボブの不正を高い確率で防止するための公平なランサムウェアプロトコルについて考えます。また、このプロトコルは Bitcoin の性質と暗号技術を巧みに利用しているため、信頼できる第三者を必要としません。なお、本稿は Qiita で公開した記事 [1] を大幅に加筆・修正したものです。

持ちます。今、ボブはアリスから貰った 1 BTC[*8]をチャーリーへ送金しようとしています。アリスからボブへのトランザクションを `Tx.1` として、またボブからチャーリーへのトランザクションを `Tx.2` としたとき、トランザクション `Tx.2` が受理されるための条件は次のようになります。

> `Tx.2` の *scriptSig* を実行し、その後スタックの状態を引き継いで `Tx.1` の *scriptPubKey* を実行し、最終的なスタックの状態が $\boxed{0}$ でなければ受理する。

これを図で表すと次の図 1 のようになります。

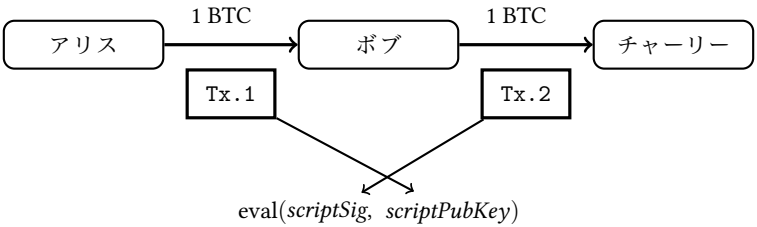

図 1: Bitcoin のトランザクションとスクリプト

そして、*scriptSig* と *scriptPubKey* を実行（eval）した結果が $\boxed{0}$ でなければ、`Tx.2` は受理されブロックチェーンに追加されます。上記の図のように、アリスからボブへ送金された 1 BTC をボブがチャーリーへの送金する場合を考えます。また、ボブの Bitcoin アドレスに対応する公開鍵を $\mathcal{B}$ とします[*9]。送金において、`Tx.2` の *scriptSig* には送金者ボブの署名 $\mathbb{S}$ と公開鍵 $\mathcal{B}$ が次のように入ります。

```
𝕊
ℬ
```

これによりスタックの状態は $\boxed{\mathcal{B}\,|\,\mathbb{S}}$ となり、この状態でアリスからボブへの送金を示すトランザクション `Tx.1` の *scriptPubKey* が実行されます。この *scriptPubKey* は次のようになります。

```
OP_DUP
OP_HASH160
h
```

[*8] “BTC” は Bitcoin の単位です。

[*9] Bitcoin アドレスは署名用の公開鍵を SHA-256 でハッシュ値を計算し、そのハッシュ値を RIPEMD-160 でもう一度ハッシュ値を計算した後、人間に読みやすいように加工を加えたものです。

urandom vol.4

発行者	urandom
表紙デザイン	秋弦めい
発行日	2017 年 8 月 13 日
バージョン	1.00（2017-08-08 01:44:53+09:00）
連絡先	https://blog.urandom.team/
印刷所	株式会社栄光