HTTPS 利用時のパスワード奪取攻撃と攻撃ツールの実装

初谷 良輔† 鬼頭 利之†† 古志 智也† 齋藤 孝道†
† 東京工科大学 ††(株) 東芝 研究開発センター

1 はじめに

SSL (Secure Socket Layer)[1] では, サーバ認証方式と相互認証方式*の 2 つの認証方式が主に利用されている. ただし, SSL の相互認証方式の利用には, サーバとクライアントの双方で CA (Certificate Authority) によって発行してもらった証明書の取得が必要不可欠であり, これにより発生する管理コストなどを考慮すると, ユーザやサービス提供者にとって容易な利用形態とはいえない. そこで, SSL のサーバ認証方式と Basic 認証を組み合わせて相互認証を実現することがある. このような利用形態において Basic 認証で用いるパスワードなどの秘密情報は, SSL の機能により暗号化されるため安全であるという見解が一般的である [2].

しかしながら, SSL のサーバ認証方式と Basic 認証を組み合わせた利用形態であったとしても, パスワードのような秘密情報が第三者によって奪取される可能性があることがわかった.

本論文では, このような認証プロトコルの組み合わせがもたらすシステム全体の脆弱性によって, パスワードなどの秘密情報が第三者に奪取されるシナリオについて示す. また, 作成した攻撃ツールを利用し, あるシナリオにおいて実際にパスワードが奪取可能であることを示す.

2 準備

本論文を通して用いる表記, 用語, および, Basic 認証の手続きについて記述する. ここで示す以外の表記, 用語, 安全性の基礎概念は, 文献 [3] に従う.

2.1 表記と用語

主体に関する表記と用語

SSL は, S で示される SSL サーバと C で示される SSL クライアントから構成される. また, 悪意ある第三者, つまり, 攻撃者は $\mathcal{I}$ で示し, サーバ S に成りすましている攻撃者を $\mathcal{I}(S)$ で, クライアント C に成りすましている攻撃者を $\mathcal{I}(C)$ で示す. ただし, C, S を通報パラメータとして利用することもあり, これは通報の生成者を特定するための識別子を示す.

通報に関する表記

Msg は, 任意の通報を表している. C が生成した Msg を S に対して送信する場合, 以下のように表現する:

$$C \to S \ : \ Msg_C$$

通報パラメータの詳細

SA は, 各主体のプロトコルバージョンとサポートする鍵交換アルゴリズム, 圧縮アルゴリズム, バルク暗号アルゴリズム, ハッシュ関数の組み合わせである暗号スイートである. RN は, ある主体がこの通報を生成した日付時刻情報と 28 バイトの乱数からなる構造体を示す. S_ID は, 各主体が保持しているセッション識別子を示し, $cert\_list$ は, 証明書チェーンを示す. pre_master_secret は, 46 バイトの乱数と 2 バイトのプロトコルバージョンから構成されている. $message\_all$ は, ある通信主体間でこの通報までに交換されたすべての通報を示す. Ack は, ある主体が SSL のセッションの確立に必要な通報を送信し終えたことを示す通報である.

$h(Msg_1)$ は, Msg_1 から生成されるハッシュ値を表す. また, $h(Msg_1, Msg_2, \cdots, Msg_i)$ と表記した場合, これは, $Msg_1, Msg_2, \cdots, Msg_i$ を結合したハッシュ値を表す.

また, あるユーザを認証するためのユーザ ID とパスワードなどの秘密情報をそれぞれ $UserID$ と $password$ で示す.

暗号化と鍵

公開鍵は P で示し, 例えば, C の公開鍵 P_C で Msg 暗号化したものを $\{Msg\}_{P_C}$ と示す. また, KS はある通信主体間で共有されるセッション鍵であり, pre_master_secret と RN から生成される. 例えば, C と S が共有するセッション鍵は, KS_{CS} となる.

2.1.1 パスワードを用いたユーザ認証

本論文で示している SSL のサーバ認証方式と組み合わせた Basic 認証とは, SSL のサーバ認証方式の手続きを終えた後, 以下のように KS_{CS} で保護したパスワードを送信することによってサーバがユーザを認証するものである:

$$C \to S \ : \ \{UserID_C, password_C\}_{KS_{CS}}$$

3 SSL サーバ認証方式 + Basic 認証の脆弱性

ここでは, 下記で定義する環境と攻撃シナリオにより秘密情報が奪取可能であることを示す. また, このことは新たに作成した攻撃ツールを用いて確認した.

3.1 前提

本節では, 主体の構成と攻撃シナリオを記す.

3.1.1 主体構成

実験環境を構成する主体は, 以下の通りである:

サーバ

正規のサーバ S であり, このマシン上では, Web サーバプログラムとして Apache-1.3.27-13 が稼動している. このサイトの index.html ファイルには, Basic 認証機能を用いてアクセス制御が掛けられたページへのリンク情報が存在する. Basic 認証を実現する手法として, Apache の認証機能, PHP, JavaScript, Servlet & JSP などがあるが, ここでは Apache の Basic 認証機能を利用している. また, この Basic 認証の手続きには, SSL を利用する.

攻撃者

攻撃者 $\mathcal{I}$ として振舞うマシンであり, このマシン上では, OpenSSL-0.9.6c-2 を用いて新たに作成した攻撃用プログラムが稼動している. また, 攻撃者が保持する証明書は, クライアントが信頼するいずれかの CA から発行されたものである.

クライアント

正規のクライアント C であり, 一般的な Web ブラウザが利用可能なマシンである. クライアントは, 事前にサーバとの間で $UserID_C$ と $password_C$ の共有を済ませている.

3.1.2 攻撃シナリオ

パスワード奪取を実現するシナリオとして, クライアントが**非自発的に攻撃者に接続**するパターン (以降, シナリオ 1) とクライアントが**自発的に攻撃者に接続**するパターン (以降, シナリオ 2) の 2 つがある.

シナリナ 1 では, 正規のサーバが管理する index.html

A Way of Compromising Password Based Authentication over HTTPS

† Ryosuke HATSUGAI (u00b155@ess.teu.ac.jp)
†† Toshiyuki KITO (toshiyuki.kito@toshiba.co.jp)
† Tomoya KOSHI (y030160e@gss.teu.ac.jp)
† Takamichi SAITO (saito@teu.ac.jp)
Tokyo University of Technology (†)
Toshiba Corporation, Crporate Research & Development Center (††)

* クライアント認証方式とも呼ばれている.

ファイル, もしくは, その他のページに予め改ざんが施されているものとする. この改ざんされたリンク情報に従いクライアントがページの取得要求を行うと, クライアントは非自発的に攻撃者へ接続し, 直ちに SSL のサーバ認証方式を開始する. その後, Basic 認証を行い, リンク先のページをクライアントは攻撃者を介して取得する (図 1).

シナリオ 2 は, 「正規のサーバのミラーサーバである」といった偽りの告知に騙されたクライアントが, 正規のサーバのように振舞っている攻撃者に自発的に接続するものである. その後, シナリオ 1 と同様に SSL のサーバ認証方式と Basic 認証の手続きを踏む.

ただし, 紙面の都合上, 本論文ではシナリオ 1 における攻撃方法だけを記す.

3.2 シナリオ 1 に基づく攻撃

3.2.1 SSL サーバ認証方式 + Basic 認証への攻撃

シナリオ 1 において, 攻撃者 $\mathcal{I}$ は, 以下の手順を踏むことにより SSL のサーバ認証方式に対して MITM (Man-In-The-Middle) 攻撃を仕掛けることが可能である. ただし, SSL には鍵交換方式として, RSA, Diffie-Hellman, Fortezza などが用意されているが, 本論文では, どの鍵交換方式を用いても本質的に同じなので, RSA の例だけを示す:

M1) $C \to \mathcal{I}(S)$: SA_C, S_ID_C, RN_C
M1′) $\mathcal{I}(C) \to S$: $SA_{\mathcal{I}}, S\_ID_{\mathcal{I}}, RN_{\mathcal{I}}$
M2) $S \to \mathcal{I}(C)$: SA_S, S_ID_S, RN_S
M2′) $\mathcal{I}(S) \to C$: $SA_{\mathcal{I}}, S\_ID_{\mathcal{I}}, RN_{\mathcal{I}}$
M3) $S \to \mathcal{I}(C)$: $cert\_list_S$
M3′) $\mathcal{I}(S) \to C$: $cert\_list_{\mathcal{I}}$
M4) $S \to \mathcal{I}(C)$: Ack_S
M4′) $\mathcal{I}(S) \to C$: $Ack_{\mathcal{I}}$
M5) $C \to \mathcal{I}(S)$: $\{pre\_master\_secret_C\}_{P_{\mathcal{I}}}$
M5′) $\mathcal{I}(C) \to S$: $\{pre\_master\_secret_{\mathcal{I}}\}_{P_S}$
M6) $C \to \mathcal{I}(S)$: $\{h(KS_{C\mathcal{I}}, h(message\_all_{C\mathcal{I}}, C, KS_{C\mathcal{I}}))\}_{KS_{C\mathcal{I}}}$
M6′) $\mathcal{I}(S) \to C$: $\{h(KS_{C\mathcal{I}}, h(message\_all_{C\mathcal{I}}, S, KS_{C\mathcal{I}}))\}_{KS_{C\mathcal{I}}}$
M7) $\mathcal{I}(C) \to S$: $\{h(KS_{\mathcal{I}S}, h(message\_all_{\mathcal{I}S}, C, KS_{\mathcal{I}S}))\}_{KS_{\mathcal{I}S}}$
M7′) $S \to \mathcal{I}(C)$: $\{h(KS_{\mathcal{I}S}, h(message\_all_{\mathcal{I}S}, S, KS_{\mathcal{I}S}))\}_{KS_{\mathcal{I}S}}$
M8) $C \to \mathcal{I}(S)$: $\{UserID_C, password_C\}_{KS_{C\mathcal{I}}}$
M8′) $\mathcal{I}(C) \to S$: $\{UserID_C, password_C\}_{KS_{\mathcal{I}S}}$

SSL のセッションの確立に必要なパラメータの交換を行う (**M1** ∼ **M4′**). その後, pre_master_secret の交換を済ませ, 各主体でセッション鍵を生成する (**M5, M5′**). ここまでのメッセージの交換を終えた $\mathcal{I}$ は, $KS_{C\mathcal{I}}$ と $KS_{\mathcal{I}S}$ の 2 つのセッション鍵を保持することになる. つまり, $\mathcal{I}$ はそれ以降に交換されるすべてのメッセージ (**M6** ∼ **M8′**) を解読, または, 生成することが可能となる. 結果として, (**M8**) において $\mathcal{I}$ は C から送信された $UserID_C$ と $password_C$ を奪取することができ, それを S に送信することで S が提供するサービスを享受することができる. さらに, $\mathcal{I}$ はそれを C に送信することで S の振りをすることもできる.

3.2.2 実際の攻撃例

上記の攻撃手順を踏まえ, シナリオ 1 においてこの攻撃を実現するための各主体の関係を図 1 に示す. また, 実際にやりとりされる HTTP ヘッダの一部を記す. なお, 図 1 における破線は, SSL を用いた通信を示している:

① C から S に対して index.html の取得要求を行う:
`GET /index.html HTTP/1.1`

② S は C からのファイル取得要求に対する応答として, index.html のリソースを送信する:
`HTTP1.1 304 Not Modified`

③ S から取得した index.html ファイルの改ざんされたリンク情報に従い C は, SSL を利用してページ取得要求

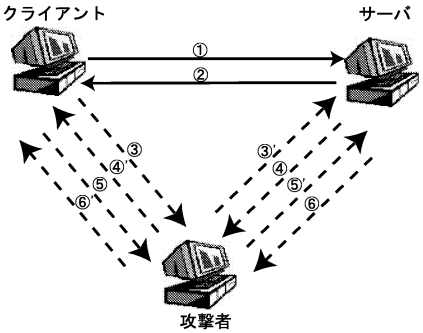

図 1: 攻撃シナリオ 1

を行う. この動作をきっかけに C は, $\mathcal{I}(S)$ と接続し, SSL のセッションを確立する:
`GET /basic_auth/test.html HTTP/1.1`

③′ $\mathcal{I}(C)$ は, C から取得した通報を SSL を利用して S に対して送信する. これをきっかけに $\mathcal{I}(C)$ と S は, SSL のセッションを確立する:
`GET /basic_auth/test.html HTTP/1.1`

④ S は, $\mathcal{I}(C)$ からのページ取得要求に対して, ステータスコード 401 を含んだ応答を返し, 要求したページの取得には, ユーザ ID とパスワードが必要であることを $\mathcal{I}(C)$ に通知する:
`HTTP/1.1 401 Authorization Required`

④′ $\mathcal{I}(S)$ は, S から受信した通報をそのまま C に送信する:
`HTTP/1.1 401 Authorization Required`

⑤ $\mathcal{I}(S)$ からの通報を受信した C は, $\mathcal{I}(S)$ に対してユーザ ID とパスワードを含んだ通報を送信する:
`GET /basic_auth/test.html HTTP/1.1`
`Authorization: Basic (`$UserID : Password$`)`

⑤′ C からの通報を受信した $\mathcal{I}(S)$ は, ユーザ ID とパスワードを奪取し, その後, この秘密情報を含んだ通報を S に送信する:
`GET /basic_auth/test.html HTTP/1.1`
`Authorization: Basic (`$UserID : Password$`)`

⑥ $\mathcal{I}(C)$ から秘密情報を受信した S はこれを確認した後, ユーザの認証に成功したことを示す通報を $\mathcal{I}(C)$ に送信する:
`HTTP/1.1 200 OK`

⑥′ $\mathcal{I}(C)$ は, S からの通報をそのまま C に送信する:
`HTTP/1.1 200 OK`

4 まとめ

本論文では, SSL のサーバ認証方式と Basic 認証を組み合わせて利用した場合, SSL のサーバ認証方式の認証プロトコルの問題によって, パスワードなどの秘密情報が第三者に奪取される可能性について示した. これは, SSL のサーバ認証方式を用いて確立された暗号化通信路が安全であるとはいえないにも関わらず、その通信路に対してクライアントがパスワードのような秘密情報を平文で送信していることに起因している. また, 定義した攻撃シナリオに従い, OpenSSL を利用して作成した攻撃ツールを実行し, 実際に秘密情報が奪取可能であることを示した.

参考文献

[1] A.Freier, P.Kocher, and P.Kaltorn, "The SSL Protocol Version 3.0 draft", March 1996, http://home.netscape.com/eng/ssl3/draft302.txt

[2] Richard E. Smith 著, "認証技術　パスワードから公開鍵まで", オーム社, 2003

[3] 齋藤孝道, "認証プロトコルの基礎", コンピュータソフトウェア (日本ソフトウェア科学会論文誌), (チュートリアル), Vol.19, No.4, pp.52-63, 2002