◇◇◇◇◇◇◇◇◇◇◇
解説 (研究紹介)
◇◇◇◇◇◇◇◇◇◇◇

SSH 通信におけるユーザ認証方式の分析と識別

佐藤　彰洋[1]
中村　豊[2]
池永　全志[3]

1　はじめに

私たちの生活はウェブやメールに代表される様々な情報サービスによって支えられています．ユーザに絶え間なく情報サービスを提供し続けるためには，管理者による対象の機器群の管理，すなわち状態の把握と異常の発見が不可欠となります．現在では，それら機器群がデータセンターなどの物理的に離れた位置に設置してあることも珍しくなく，遠隔からの機器の操作を実現する Secure Shell (SSH) が管理者にとって必須のツールとなっています．SSH は，機器上で動作する SSH サーバと管理者が操作する端末上で動作する SSH クライアントで相互に通信することで，管理者による遠隔からの機器の操作を実現します．

2010 年に，米国の情報セキュリティに関する研究機関である SysAdmin, Audit, Network, Security Institute (SANS) は，SSH を標的とした総当り攻撃の急増に対する注意勧告を発表しました [1]．SSH 総当り攻撃とは，攻撃者の保持する辞書に基づいた膨大な量の試行により，ユーザのパスワードを奪取する攻撃です．パスワードが奪取されると，機密情報の漏洩，フィッシングサイトの構築，スパムの送信などの深刻な事態を招く恐れがあるため，その攻撃への対策は管理者にとって急務になっています．

従来，ネットワークで観測されるフローから総当り攻撃の有無を判断してきました [2, 3]．具体的には，機器の遠隔操作などに代表される手動処理の通信 [4] が長期に渡ること，総当り攻撃の通信が短時間に接続と切断を繰り返すことに着目し，それらの差異を利用しています．しかしながら，機器における自動処理に SSH を用いる場合 [5, 6]，それらの通信と総当り攻撃の通信が類似していること原因で，既存手法による正確な検出が困難となります．

本稿では，個々の通信におけるユーザ認証方式を識別することで上述の問題の解決を図ります．その理由は，(1) 総当り攻撃がキーストローク型認証を標的とした攻撃であること，(2) SSH を通じた自動処理には非キーストローク型認証が不可欠であることに起因します．キーストローク型とは，ユーザにより入力された文字列のみに基づいて正当性を判断する方式です．一方，非キーストローク型とは，文字列以外の情報に基づいて正当性を判断する方式であり，ユーザが介在しない認証を実現できます．従って，後者の非キーストローク型認証を利用する通信を除外することで，総当り攻撃の検出精度の向上が期待できます．

2 節で SSH に関連する技術について述べた後，総当り攻撃の検出における問題点を整理します．3 節の分析結果に依拠して，4 節で個々の通信におけるユーザ認証方式を識別するための手法を提案します．5 節で提案手法を評価し，6 節で本稿をまとめます．

[1]情報科学センター　助教　satoh@isc.kyutech.ac.jp
[2]情報科学センター　准教授　yutaka-n@isc.kyutech.ac.jp
[3]大学院工学研究院電気電子工学研究系システムエレクトロニクス部門　教授　ike@ecs.kyutech.ac.jp

2 関連技術

本節では，先ず SSH のサブプロトコルとユーザ認証方式を説明します．次いで，SSH に関する既存研究について述べた後，総当り攻撃の検出における問題点を整理します．

2.1 SSH の仕様

SSH は 3 種類のサブプロトコル，すなわちトランスポート層プロトコル [7]，ユーザ認証プロトコル [8]，コネクションプロトコル [9] により構成されます．以降，各サブプロトコルの詳細について説明します．

トランスポート層プロトコルは，高信頼・高安全を保証する通信路をクライアント・サーバ間で確立します．データの盗聴・改竄の防止，及び高効率なデータの転送を実現するため，クライアント・サーバ間のネゴシエーションで，暗号，MAC (Message Authentication Code)，圧縮の各アルゴリズムを決定します．留意すべきは，これ以降にクライアント・サーバ間で送受信されるデータには，前述のアルゴリズムが適用される点です．

ユーザ認証プロトコルは，トランスポート層で確立した通信路の利用の可否を認証によって判断します．まず，そのサブプロトコルの開始に関する情報を交換した後に，ユーザ認証方式によりユーザの正当性を評価します．ユーザ認証方式の例は公開鍵認証やパスワード認証などであり，その詳細については 2.2 節で述べます．

コネクションプロトコルは，通信路の確立と認証が行われた後に，対話型セッション，ファイル転送，および各種フォワーディングなどのサービスをユーザに提供します．トランスポート層プロトコルで確立した通信路を多重化することにより，論理的なチャネルを作成します．その後，各チャネルに対してユーザが要求するサービスが割り当てられます．

SSH の有する重要な性質として，機密性と柔軟性が挙げられます．機密性とは，暗号アルゴリズムとメッセージ認証アルゴリズムで実現される，攻撃者による盗聴・改竄からデータを保護する機能です．また柔軟性とは，クライアントとサーバのネゴシエーションにより，暗号アルゴリズム，MAC アルゴリズム，圧縮アルゴリズムを採択する機能です．

2.2 ユーザ認証方式

ユーザ認証方式とは，SSH を用いて接続を要求してきた通信に対して，ユーザの正当性を確認する手段です．一般的な SSH の実装である OpenSSH [10] と Tectia [11] において，共通して採用されているのは，(1) パスワード認証，(2) チャレンジ・レスポンス認証，(3) 公開鍵認証，(4) ホストベース認証，(5) Kerberos 認証です．本稿では，最後を除外した 4 種類のユーザ認証方式を対象とします．その理由として，Kerberos を利用した認証は実現方法が多岐に渡るため，その個々について検証することが困難な点が挙げられます．

ユーザ認証方式は，キーストローク型と非キーストローク型に分類できます．キーストローク型は，文字列の正誤のみから正当性を判断する方式であり，パスワード認証，およびチャレンジ・レスポンス認証が該当します．この認証は，機器の遠隔操作などに代表される手動処理の通信で利用されます．一方，非キーストローク型は，文字列以外の情報に基づいて正当性を判断する方式です．例えば，公開鍵認証では秘密鍵と公開鍵の対，ホストベース認証では機器間の信頼関係を利用されます．これらはユーザが介在しない認証が実現可能であるため，手動処理に加え自動処理にも適用されます．

2.3 既存研究と問題点

これまでに SSH 総当り攻撃に関する様々な研究が行われてきました．文献 [12, 13] では，総当り攻撃の種類や傾向，および攻撃成功後の動作について報告されています．他には，擬似的なトラフィックの生成を目的とした総当り攻撃のモデル化 [14]，トラフィックの効率的な分析を目的としたフローの関係性の可視化 [15] などが挙げられます．

SSH 総当り攻撃の検出を目的とした研究は，(1) アクセスログに基づく手法 [16, 17]，(2) トラフィックに基づく手法 [2, 3] に大別されます．前者は，個々の機器においてアクセスログから認証の成否を計測し，その頻度が閾値を超えた場合に攻撃と判断します．しかしながら，比較的規模の大きい組織では，そのネットワークに点在する全機器のアクセスログを確認する必要がるため，その運用には多大な労力が必要となります．後者は，SSH による手動処理の通信 [4] が長期に渡ること，総当り攻撃の通信が短時間に接続と切断を繰り返すことに着目し，それらの差異から攻撃の有無を判断します．この方法は，外部との接続点におけるトラフィックの計測のみで総当り攻撃を検出できるため，その運用に要する労力を大幅に低減できます．しかしながら，SSH による自動処理 [5, 6] が頻発するネットワークでは，それらの通信と総当り攻撃の通信との類似性が原因で，既存手法による検出精度の低下が問題となります．

3 分析

本稿では，各通信のユーザ認証方式を識別することで 2.3 節で述べたの問題の解決を図ります．その理由は，(1) 総当り攻撃がキーストローク型認証を標的とした攻撃であること，(2) SSH を通じた自動処理には非キーストローク型認証が不可欠であることに起因します．従って，後者の非キーストローク型認証を利用する通信を除外することで，SSH 総当り攻撃の検出精度の向上が期待できます．

個々の通信におけるユーザ認証方式を識別するにあたり，障害となるのが SSH の機密性と柔軟性です．機密性とは，攻撃者による盗聴・改竄からデータを保護する機能であり，故に，ユーザ認証方式を直接調査できないことが問題となります．本手法における第一の特色は，ユーザ認証方式により送受信される情報が異なることに着目し，その問題をフローの挙動を用いることで解決することにあります．フローとは，パケットの送信元および送信先アドレスやポート番号，プロトコル番号の組に基づいて識別可能な，クライアント・サーバ間で行われる双方向の通信となり，その挙動とは，フローを構成する個々のパケットから計測可能な，パケットのサイズ，通信方向，および到着順などの統計値となります．また柔軟性とは，クライアント・サーバ間のネゴシエーションで暗号，MAC，圧縮の各アルゴリズムを採択する機能であり，故に，その決定でフローの挙動が変化することが問題となります．第二の特色は，それら変化を相殺するための基準点を考慮してフローの挙動を導出することにあります．その基準点には，各アルゴリズムが適用済みであること，保持する情報がユーザ認証方式に依存しないことから，ユーザ認証プロトコルにおける先頭のパケットを利用します．

本節では，機密性と柔軟性から生じる問題に対する提案手法の有効性を分析により示します．3.1 節で，分析に用いるデータセットについて述べます．3.2 節と 3.3 節で，各通信におけるユーザ認証方式が識別可能であることを明らかにします．

3.1 データセット

分析に用いたデータセットを表 1 に示します．これらのフローは，表 2 の全ての組合せで，ユーザ認証方式，および各種アルゴリズムを変更することにより生成しました．また，$D1$ と $D2$ の違いはフローにおける認証の成否です．ここで，公開鍵認証の失敗とは，非正規の鍵を用いて認証を試みた場合であり，ホストベース認証の失敗とは，未登録の機器から接続を試みた場合です．

表 1: 分析用データセットにおけるフローの数.

	password	chal-resp	public-key	host-based
$D1$	182	182	546	546
$D2$	182	182	546	546

表 2: ユーザ認証方式，暗号アルゴリズム，MAC アルゴリズム，圧縮アルゴリズムの種類.

User Auth. Method	password, challenge-response, public-key (rsa, dsa, ecdsa), host-based (rsa, dsa, ecdsa)
Cipher Algorithm	aes128-ctr, aes192-ctr, aes256-ctr, arcfour256, arcfour128, aes128-cbc, 3des-cbc, blowfish-cbc, cast128-cbc, aes192-cbc, aes256-cbc, arcfour, rijndael-cbc@lysator.liu.se
MAC Algorithm	hmac-md5, hmac-sha1, hmac-md5-96, hmac-sha1-96, hmac-ripemd160, umac-64@openssh.com, hmac-ripemd160@openssh.com
Compression Algorithm	none, zlib

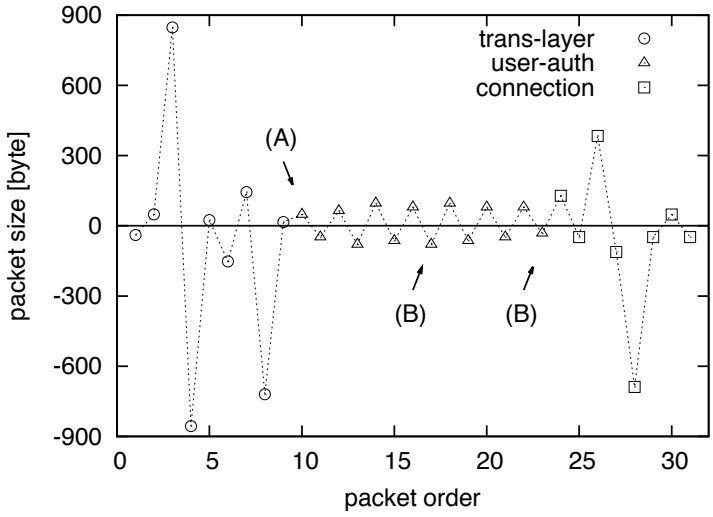
留意するべき点としては，これらのデータセットは全ての TCP 制御パケットを除外していることです．TCP 制御パケットとは，TCP ペイロードが存在せず，Ack や Syn，Fin などの情報のみを持つ，通信の制御を目的としたパケットです．除外の理由は，上位層のプロトコルである SSH が TCP 制御パケットに影響を及ぼさないことから，ユーザ認証方式の識別に必要な情報を TCP 制御パケットが保持しないためです.

3.2 ユーザ認証プロトコルにおける通信に関する分析

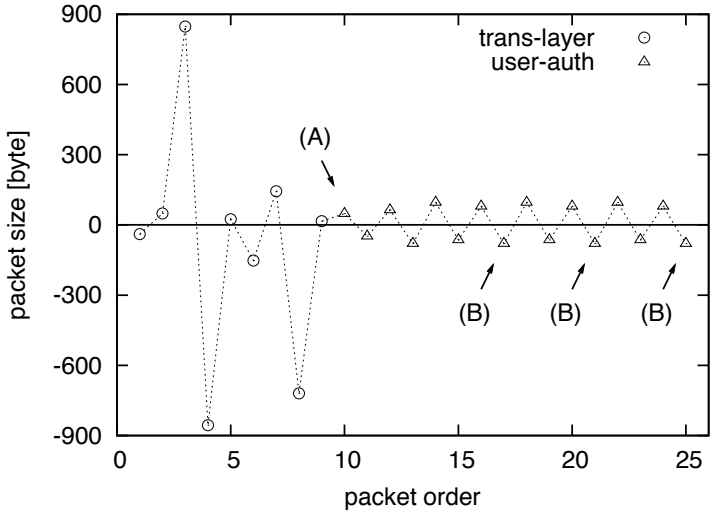
本節では，個々のフローにおけるユーザ認証方式の識別を実現するための前準備として，ユーザ認証プロトコルにおいてクライアント・サーバ間で送受信されるパケットの詳細を調査します.

データセット $D1$ と $D2$ から任意に選択したフローを対象に，それらを構成する個々のパケットについて，(i) パケットの到着順，(ii) パケットサイズ，(iii) パケットの通信方向，(iv) パケットの種類を計測し，文献 [18] の手法に基づいて，それらの関係の可視化を行いました．その結果の一部を図 1 に示します．X 軸はパケットの到着順，Y 軸における値の大きさはパケットサイズ，その正負はパケットの通信方向を表します．正の場合は，クライアントからサーバへの要求である “incoming”，負の場合は，サーバからクライアントへの応答である “outgoing” を意味します．例えば $(10, -500)$ の点は，フローの先頭から 10 番目，且つ “outgoing” の向きに送信された，データサイズが $500byte$ のパケットとなります．また，プロットとラベルにより，パケットの種類を表現しています．具体的には，プロットの種類は，各パケットが属するサブプロトコルを，付与されているラベルは，(A) 暗号，MAC，圧縮の開始箇所，(B) 認証結果の通知箇所を意味します.

図 1 から，ユーザ認証プロトコルにおける通信は，要求と応答のパケットの繰り返しで構成されることが見て取れます．また，そのサブプロトコルの開始直後に，暗号，MAC，圧縮の各アルゴリズムが適用されていることが解ります．さらに注目すべきは，そのフローの最後に位置する認証において，成功を通知した直後に次のサブプロトコルに推移する点，失敗を通知した直後に通信の切断が発生する点



(a) チャレンジ・レスポンス認証の成功



(b) チャレンジ・レスポンス認証の失敗

図 1: 各サブプロトコルにおけるパケットの交換.

です.

以上の結果から，ユーザ認証プロトコルは，(1) 要求と応答で構成されていること，(2) 開始直後に各アルゴリズムが適用されること，(3) 認証の成否で以降の動作がことなることが明らかになりました.

3.3 ユーザ認証方式の識別に関する分析

本節では，ユーザ認証方式の種類による通信の差異を明らかにするために，データセット $D1$ と $D2$ を対象としてパケットの特徴を調査します.

分析をするにあたり，フローの挙動の新たな値としてパケットの特徴を定義します．パケットの特徴

とは，(i) パケットのサイズ，(ii) パケットの通信方向を考慮して，式 1 により導出される値です．

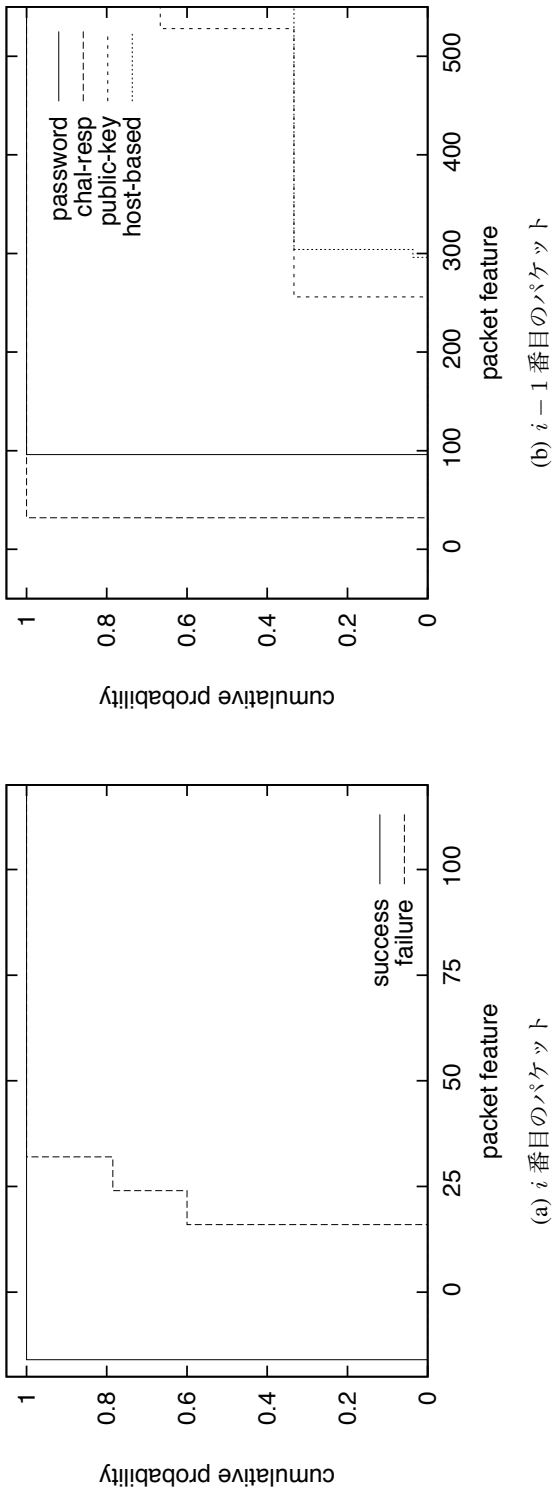
$$f_i(x) = \begin{cases} s_i(x) - s_n(x) & \big(\ d_i(x) = \text{incoming}\ \big), \\ s_i(x) - s_{n+1}(x) & \big(\ d_i(x) = \text{outgoing}\ \big). \end{cases} \tag{1}$$

ここで，$s_i(x)$ は，フロー x における i 番目のパケットのサイズ，$d_i(x)$ は，フロー x における i 番目のパケットの通信方向です．具体的には，“incoming” はクライアントからサーバへの要求，“outgoing” はサーバからクライアントへの応答を意味します．また，ユーザ認証プロトコルのおける先頭の要求が n 番目のパケット，その返答が $n+1$ 番目のパケットとします．留意すべきは，ユーザ認証プロトコルにおける先頭の要求と応答は，(1) そのサブプロトコルの開始に関連する情報のみを保持するためユーザ認証方式に依存しない点，(2) 暗号，MAC，圧縮の各アルゴリズムが適用されている点です．従って，それらパケットを基準とすることで，フローの挙動に対する各アルゴリズムの影響を除外できます．

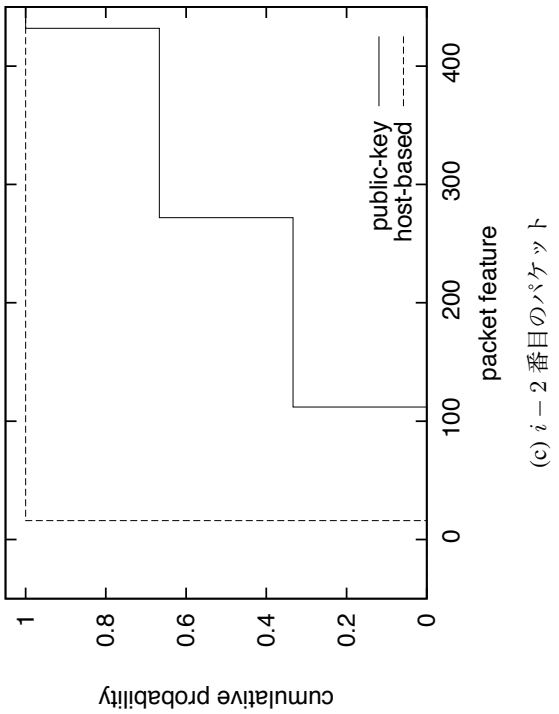
分析の流れは次の通りです．始めに，データセット $D1$ と $D2$ における全フローを対象に，認証結果の通知箇所を起点として部分フローを選択しました．部分フローとは，フローから切り出される連続したパケットの系列です．ここでは，認証結果の通知箇所が i 番目のパケットである場合，$i-2$ 番目から i 番目までのパケットが部分フローとして選択します．次に，式 1 に基づいて，それら部分フローからパケットの特徴を導出しました．最後に，それらパケットの特徴を視覚的に比較するために，それら計測値を図 2 の累積密度分布で示しました．その X 軸はパケットの特徴，Y 軸は累積密度を表します．

図 2(a) から，i 番目のパケットの特徴は，認証の成否のみの影響を受けることが見て取れます．これは，認証結果として通知される情報がユーザ認証方式の種類に依存しないことが原因です．図 2(b) から，パスワード認証とチャレンジ・レスポンス認証のフローの特徴が，それぞれ 100 付近の値，30 付近の値であることが解ります．しかしながら，公開鍵認証とホストベース認証では 250 から 650 と，フローの特徴が非常に類似した値を示しています．一方，図 2(c) から，公開鍵認証が 120 から 450，ホストベース認証が 15 付近の値と，それぞれ異なる範囲を取ることが解ります．図 2(b) と図 2(c) から読み取れるパケットの特徴の違いは，ユーザ認証方式によって送受信される情報が異なることが原因です．

以上の結果から，部分フローから導出されるパケットの特徴が，認証の成否のみに依存する箇所と各ユーザ認証方式で差異がある箇所に大別できることが明らかになりました．



(a) i 番目のパケット　(b) $i-1$ 番目のパケット



(c) $i-2$ 番目のパケット

図 2: パケットの特徴の累積密度分布.

4 提案

3.2 節と 3.3 節の分析結果に基づいて，個々の通信におけるユーザ認証方式を識別するための手法を提案します．本手法は，(1) 基準パケット選択機能，(2) パケット特徴導出機能，(3) ユーザ認証手法識別機能により構成されます．以降，それぞれの機能の詳細について述べます．

4.1 基準点選択機能

本機能は，パケットの特徴を導出するにあたり必要となる基準点を選択します．その実現のために，基準点となるパケットが，暗号，MAC，圧縮の各アルゴリズムが適用される最初の要求と応答である点を利用します．

フロー x において，パケットの内容を逐次調査することにより，その暗号化の有無を判別します．具体的には，$c_n(x)$ と $c_{n+1}(x)$ が真，且つ $c_{n-1}(x), \cdots, c_1(x)$ が偽である場合，$p_n(x)$ と $p_{n+1}(x)$ のパケットを基準点として選択します．ここで，$p_n(x)$ は，フロー x における n 番目のパケット，$c_n(x)$ は，フロー x における n 番目のパケットの暗号化の有無を意味します．

4.2 パケット特徴導出機能

本機能は，(1) パケットの通信方向に基づく部分フローの検出と，(2) ユーザ認証プロトコルの末尾に位置する部分フローの判別により，(3) ユーザ認証方式の識別に必要なパケットの特徴を導出します．

まず，ユーザ認証プロトコルの先頭から順にパケットの通信方向を調査することで，要求と応答の繰り返しを部分フローとして検出します．具体的には，$d_i(x) = outgoing$，$d_{i-1}(x) = incoming$，且つ $d_{i-2}(x) = outgoing$ の条件を満たすとき，連続するパケット $p_{i-2}(x), p_{i-1}(x), p_i(x)$ が部分フロー x_i となります．ここで，$d_i(x)$ は，フロー x における i 番目のパケットの通信方向を意味します．

次いで，その部分フローがユーザ認証プロトコルの末尾に位置するか否かを判別します．その判別には，(1) 目的の箇所に位置する部分フローが認証の成否を保持する点，(2) 認証が成功した直後に次のサブプロトコルに推移する点，(3) 最後の認証が失敗した直後に通信の切断が発生する点を利用します．具体的には，$f_i(x) \in \mathbb{Z}_s^i$ を満たすとき，または $f_i(x) \in \mathbb{Z}_f^i$ 且つ $\forall j\ \neg p_j^*(x)\ s.t.\ i < j$ を満たすとき，部分フロー x_i の位置をユーザ認証プロトコルの末尾と見なします．ここで，$f_i(x)$ は，フロー x における i 番目のパケットの特徴を意味します．また，集合 $\mathbb{Z}_s^i$ は認証の成功，集合 $\mathbb{Z}_f^i$ は認証の失敗を保持するパケットの特徴が取り得る範囲を示します．

最後に，ユーザ認証方式の識別に必要なパケットの特徴を導出します．具体的には，式 1 に基づいて $p_{i-1}(x)$ と $p_{i-2}(x)$ からパケットの特徴を導出し，その結果である $f_{i-1}(x)$ と $f_{i-2}(x)$ を出力します．

4.3 ユーザ認証方式識別機能

本機能は，前機能からの入力であるパケットの特徴に基づいて，個々の通信で採用されているユーザ認証方式を識別します．

パケットの特徴が $f_{i-1}(x) \in \mathbb{Z}_p$ を満たすときパスワード認証，$f_{i-1}(x) \in \mathbb{Z}_c$ を満たすときチャレンジ・レスポンス認証を識別結果として出力します．また，$f_{i-1}(x) \in \mathbb{Z}_{kh}$ 且つ $f_{i-2}(x) \in \mathbb{Z}_k$ を満たすとき公開鍵認証，$f_{i-1}(x) \in \mathbb{Z}_{hk}$ 且つ $f_{i-2}(x) \in \mathbb{Z}_h$ を満たすときホストベース認証を識別結果として出力します．ここで各集合は，パスワード認証，チャレンジ・レスポンス認証，公開鍵認証，及びホストベース認証において，$i-1$ 番目と $i-2$ 番目のパケットの特徴が取り得る範囲となります．留意すべき点として，集合 $\mathbb{Z}_{kh}$ の対象が，公開鍵認証とホストベース認証の両方式における $i-1$ 番目のパケットの特徴であることが挙げられます．上述の条件式と出力結果の関係を，表 3 に示します．

表 3: 条件式と出力結果の関係.

condition	result
$f_{i-1}(x) \in \mathbb{Z}_p$	password
$f_{i-1}(x) \in \mathbb{Z}_c$	challenge-response
$f_{i-1}(x) \in \mathbb{Z}_{hk}$ and $f_{i-2}(x) \in \mathbb{Z}_k$	public-key
$f_{i-1}(x) \in \mathbb{Z}_{hk}$ and $f_{i-2}(x) \in \mathbb{Z}_h$	host-based

表 4: 評価用データセットにおけるフローの数.

	dictionary attacks	public-key	host-based
$D1$	0	546	546
$D2$	0	546	546
$D3$	2000	0	0

5 評価

本節では，実験を通じて，提案手法が SSH 総当り攻撃の検出精度の向上に貢献できることを示します．5.1 節でデータセットとパラメタ，5.2 節で評価指標について述べた後，5.3 節で実験結果について議論します.

5.1 緒言

表 4 に，実験に用いたデータセットにおけるフローの数を示します．公開鍵認証とホストベース認証を用いたフローは，3.1 節で述べたデータセットと同様の条件で生成しました．また，SSH 総当り攻撃は，ツール [19] により攻撃を試みることで，実際にフローを計測しました.

実験における各種パラメタは，認証の成否の判別に関する集合を $\mathbb{Z}_s = [-20, -10]$，$\mathbb{Z}_f = [10, 50]$ に，ユーザ認証方式の識別に関する集合を $\mathbb{Z}_p = [90, 100]$，$\mathbb{Z}_c = [25, 35]$，$\mathbb{Z}_{kh} = [250, 650]$，$\mathbb{Z}_k = [100, 450]$，$\mathbb{Z}_h = [0, 50]$ に設定しました．これらの最適化については今後の課題とします.

5.2 評価指標

ユーザ認証方式の識別において，その精度を評価するための指標に再現率と適合率を用います．クラス X の識別における再現率 $R(\mathbb{X})$ と適合率 $P(\mathbb{X})$ を，式 2 に示します.

$$R(\mathbb{X}) = \frac{TP(\mathbb{X})}{TP(\mathbb{X}) + FN(\mathbb{X})}, \qquad P(\mathbb{X}) = \frac{TP(\mathbb{X})}{TP(\mathbb{X}) + FP(\mathbb{X})} \tag{2}$$

ここで，$TP(\mathbb{X})$ は，真のクラスが $\mathbb{X}$ である場合にクラス $\mathbb{X}$ と識別した数，$FN(\mathbb{X})$ は，真のクラスが $\mathbb{X}$ である場合にクラス $\bar{\mathbb{X}}$ と識別した数，$FP(\mathbb{X})$ は，真のクラスが $\bar{\mathbb{X}}$ である場合にクラス $\mathbb{X}$ と識別した数です.

表 5: ユーザ認証方式の識別結果.

	public-key	host-based
R	0.995	1.000
P	0.997	1.000

5.3 議論

表 5 に，提案手法によるユーザ認証方式の識別結果を示します．その結果から，公開鍵認証とホストベース認証の良種の通信において，再現率が 0.995 以上，適合率が 1.000 と，非常に高い精度を示すことが見て取れます．再現率 0.005 程度の低下は，総当り攻撃の通信と自動処理の通信が集中したことにより，計測時にパケットの損失が発生したことが原因です．上述の問題については，今後検討する必要があります．

従来，ネットワークのスキャンにより，機器で有効なユーザ認証方式を調査してきました [20, 21]．しかしながら，SSH は数種の認証を同時に許可できるため，個々の通信で採用しているユーザ認証方式を特定することは困難でした．提案手法では，フローの挙動の導出に基準点を考慮することにより，個々の通信におけるユーザ認証方式の識別を高精度で実現しました．また，その識別結果に基づいて非キーストローク型認証の通信を除外することにより，既存の SSH 総当り攻撃検出手法の精度の向上が期待できます．

6 おわりに

本稿では，SSH の詳細な分析から得られた知見に依拠して，各通信のユーザ認証方式を識別するための手法を提案しました．また実験を通じて，提案手法により高精度での識別を実現できることを確認しました．その識別結果に基づいて非キーストローク型認証の通信を除外することにより，既存の SSH 総当り攻撃検出手法の精度の向上が期待できます．

今後は，大規模なネットワークで計測されたトラフィックを対象に，識別精度と所要時間を評価する予定です．

参考文献

[1] SANS Internet Storm Center. `http://isc.sans.edu/`.

[2] Kazuya Takemori, Dennis Arturo Ludena Romana, Shinichiro Kubota, Kenichi Sugitani, and Yasuo Musashi. Detection of NS Resource Record DNS Resolution Traffic, Host Search, and SSH Dictionary Attack Activities. *International Journal of Intelligent Engineering and Systems*, 2(4):35–42, 2009.

[3] Laurens Hellemons, Luuk Hendriks, Rick Hofstede, Anna Sperotto, Ramin Sadre, and Aiko Pras. SSHCure: A Flow-Based SSH Intrusion Detection System. *Lecture Notes in Computer Science*, 7279:86–97, 2012.

[4] Federico Pellegrin and Christian Pellegrin. System Administration: Secure Logging over a Network. *Linux Journal*, (74), 2000.

[5] John Ouellette. Paranoid Penguin: Managing SSH for Scripts and Cron Jobs. *Linux Journal*, (137), 2005.

[6] Vladislav Marinov and Jurgen Schonwalder. Performance Analysis of SNMP over SSH. *Proceedings of the 17th IFIP/IEEE International Conference on Distributed Systems: Operations and Management*, 12(3):25–36, 2006.

[7] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Transport Layer Protocol. *RFC 4253*, 2006.

[8] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Authentication Protocol. *RFC 4252*, 2006.

[9] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Connection Protocol. *RFC 4254*, 2006.

[10] OpenSSH. `http://www.openssh.com/`.

[11] Tectia. `http://www.ssh.com`.

[12] Jim Owens and Jeanna Matthews. A Study of Passwords and Methods Used in Brute-Force SSH Attacks. *Technical Report*, 2008.

[13] Daniel Ramsbrock, Robin Berthier, and Michel Cukier. Profiling Attacker Behavior Following SSH Compromises. *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 119–124, 2007.

[14] Anna Sperotto, Ramin Sadre, Pieter-Tjerk de Boer, and Aiko Pras. Hidden Markov Model Modeling of SSH Brute-Force Attacks. *Lecture Notes in Computer Science*, 5841:164–176, 2009.

[15] Hirochika Asai, Kensuke Fukuda, and Hiroshi Esaki. Traffic Causality Graphs: Profiling Network Applications through Temporal and Spatial Causality of Flows. *Proceedings of the 12th INFORMS Computing Society Conference*, pages 95–102, 2011.

[16] J. Lane Thames, Randal Abler, and David Keeling. A Distributed Active Response Architecture for Preventing SSH Dictionary Attacks. *Proceedings of the IEEE Southeast Conference*, pages 84–89, 2008.

[17] Yen-Ning Su, Guang-Han Chung, and Benjamin Jenghorng Wu. Developing the Upgrade Detection and Defense System of SSH Dictionary-Attack for Multi-Platform Environment. *Journal of iBusiness*, 3(1):65–70, 2011.

[18] Charles V. Wright, Fabian Monrose, and Gerald M. Masson. Using Visual Motifs to Classify Encrypted Traffic. *Proceedings of the 3rd International Workshop on Visualization for Computer Security*, pages 41–50, 2006.

[19] SSHBrute. `http://github.com/jbob/sshbrute/`.

[20] Niels Provos and Peter Honeyman. ScanSSH - Scanning the Internet for SSH Servers. *Proceedings of the 15th USENIX Systems Administration Conference*, pages 25–30, 2001.

[21] Net-Scan-SSH-Server-SupportedAuth.
`http://search.cpan.org/dist/Net-Scan-SSH-Server-SupportedAuth/`.