

副テーマ研究報告書
隔離ネットワークにおける通信制御の調査

1910225 嶠南 秀敏

副テーマ指導教員 知念 賢一 特任准教授
主指導教員 藤崎 英一郎 教授

2020 年 10 月 16 日

概要

現在、ユーザの遠隔サーバへの利用を容易にするアプリケーションは多く存在する。その一方で、サーバへの通信接続をシステム管理者が容易に制御管理するためのアプリケーションは少ない。そのような通信制御のアプリケーションなしでは、システム管理者はユーザーの接続ごとに手動でコマンドを実行し管理しなければならず、サーバを利用するユーザーが多くなるに従いシステム管理者への負担が多くなってしまう。

そこで、本研究ではサーバへの通信制御に有効なアプリケーションを調査することを目的とし、三台のサーバーと一台のL2スイッチを用いて隔離ネットワークを構成し、ゲートウェイサーバーに個人のソースコード投稿サイトに公開されている通信制御のアプリケーションを試用することで、通信制御の容易さ、通信の可視化度合い、セキュリティ、三点について検討を行う。

目次

第1章 はじめに	3
1.1 研究背景	3
1.2 研究目的	3
第2章 遠隔アクセス技術	6
2.1 暗号化技術	6
2.1.1 共通鍵暗号方式	6
2.1.2 公開鍵暗号方式	8
2.2 トンネリング (Tunneling)	8
2.2.1 IPSec	9
2.2.2 PPTP	10
2.2.3 L2TP	10
2.3 Telnet	11
2.3.1 Telnet の仕組み、使用法	11
2.4 SSH	12
2.4.1 SSH の機能	14
2.4.2 SSH 接続までの流れ	14
2.5 VPN	16
2.5.1 IPSec-VPN	16
2.5.2 SSL-VPN	17
第3章 認証技術	21
3.1 LDAP	21
3.2 Kerberos	23
3.3 RADIUS	26
3.4 Active Directory	27
第4章 研究のためのネットワーク構成	28
第5章 小規模ソフトウェア	31
5.1 sshuttle	31
5.2 sshportal	33
第6章 大規模ソフトウェア	35
6.1 Teleport	35
6.2 SoftEther VPN	41

6.2.1 リモートアクセス VPN (SoftEtherVPN)	42
---	----

第 7 章 通信制御に関する考察	51
-------------------------	-----------

第 8 章 まとめ	57
------------------	-----------

第1章 はじめに

1.1 研究背景

クラウドサーバーサービスの普及により、一般ユーザーが遠隔サーバーにアクセスし、その資源を利用する機会が増えてきた。Google のクラウドサービス (GCP)、Amazon のクラウドサービス (AWS)、Microsoft のクラウドサービス (Azure) などのようにユーザーの目的に合わせて、遠隔サーバーの利用を容易にするアプリケーションが多く存在する。

しかし、一方でサーバーへの通信接続をシステム管理者が容易に制御管理するためのアプリケーションは少ない。そのような通信制御のアプリケーションなしでは、システム管理者はユーザーの接続ごとに手動でコマンドを実行しなければならず、サーバーを利用するユーザーが多くなるに従いシステム管理者への負担が多くなってしまう。

令和2年現在新型コロナウィルス感染症の世界的大流行により在宅勤務(テレワーク)が推奨され、自宅から社内サーバーへのアクセスを余儀なくされている。そして、同年7月下旬に政府が在宅勤務7割を企業に要請した[1]ことにより、今後ますますテレワークの人口が増大していきそれに伴い、大規模なリモートアクセスにおけるセキュリティがより重要になり、社内のシステム管理者の負担がますます増加していくことが予想できる。

1.2 研究目的

総務省テレワークセキュリティガイドライン[2]には、テレワークの利用方法が6パターンに分類されており、それとともに対策すべき観点もまとめられている。それを以下の表1.1に示す。下表1.1によると6パターンのうち1、2、6の3パターンが外部から社内サーバにアクセスを必要とする「オンプレミス型」のもので、3、4、5の3パターンはインターネット上のクラウドサービスを利用している「オフプレミス型」である。

それぞれのパターンの細かな違いは、テレワーク利用端末にデータを保存できるかなどであり、オンプレミス型もオフプレミス型どちらもインターネットを介して、サーバにアクセスしている。

本研究では「トンネリング」や「VPN」などのテレワークや遠隔アクセスの根底にある技術の解説を行うと同時に、社内ネットワークを想定した、隔離ネットワークを構成し、そこにソースコード投稿サイトに公開されている通信制御アプリケーションを試用することで、その中で小規模と大規模アプリケーションで分類しソフトウェアの検討を行う。

本研究でまとめることで、ブラックボックス的にリモートアクセスを行うのではなく、どのような技術や仕組みを用いて実現しているか、高いセキュリティの実現のためにどのようなプロトコルを用いているかを、リモートアクセス利用者の理解を深めることを目的

とする。それと同時にシステム管理者が通信接続の制御をすることを容易にするためのアプリケーションを調査する。

表 1.1: テレワークの 6 種類のパターン

	パターン1 リモートデスクトップ方式	パターン2 仮想デスクトップ方式	パターン3 クラウド型アプリ方式	パターン4 セキュアブラウザ方式	パターン5 アプリケーションラッピング方式	パターン6 会社PCの持ち帰り方式
概要	オフィスにある端末を遠隔操作	テレワーク用の仮想端末を遠隔操作	クラウド上のアプリケーションを社内外から利用	特別なブラウザを用いて端末へのデータ保存を制限	テレワーク端末内への保存を不可とする機能を提供	オフィスの端末を持ち帰りテレワーク端末として利用
テレワーク端末に電子データを保存するか？	保存しない	保存しない	どちらも可	保存しない	保存しない	保存する
オフィスの端末と同じ環境を利用するか？	同じ	テレワーク専用の環境	クラウド型アプリに関しては同じ	ブラウザ経由で利用するアプリに関しては同じ	テレワーク専用の環境	同じ
クラウドサービスを利用するか	しない	しない	する	する	する/しないどちらも可	する/しないどちらも可
高速インターネット回線の必要性	必須	必須	望ましい	望ましい	望ましい	不要

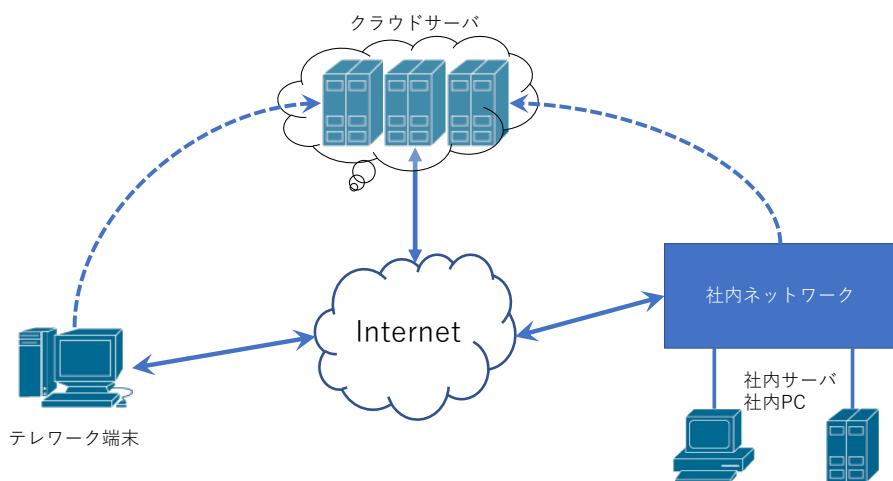


図 1.1: オフプレミス型

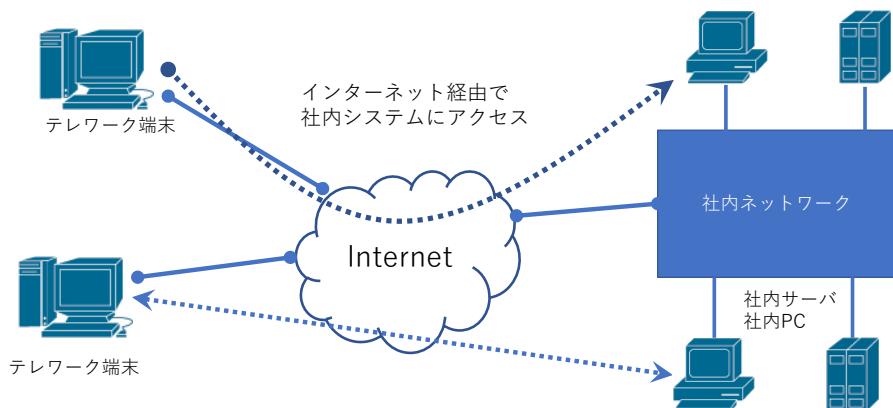


図 1.2: オンプレミス型

第2章 遠隔アクセス技術

本章では遠隔サーバーに接続し操作するための技術やソフトウェアの解説を行う。

2.1 暗号化技術

暗号化技術は、情報の保護やコンピュータセキュリティに欠かせない技術である。通信内容の保護のために、一般的に次に示す二種類の暗号化技術を使用して、認証及び暗号化通信を行っている。

- 共通鍵暗号方式
- 公開鍵暗号方式

それぞれの暗号方式は様々なアルゴリズムによって実現されるが、元となる「平文」データを「鍵」を使って「暗号文」に変換している。また、「暗号文」は「鍵」を使用して、「平文」に復号できる。

2.1.1 共通鍵暗号方式

共通鍵暗号方式は、下図 2.1 のように A と B で共通の鍵を使用して、暗号化と復号化を行う。安全にデータを暗号化するために、暗号化通信をする前に共通鍵を事前に秘密に共有する必要がある。

共通鍵暗号方式は、後述の公開鍵暗号方式に比べて、演算の処理量が少ないという利点ある。そのため、SSH プロトコルやインターネット上の通信において、通信内容の暗号化にはこの共通鍵暗号方式を採用している。

代表的な暗号化アルゴリズムに「AES」が存在する。

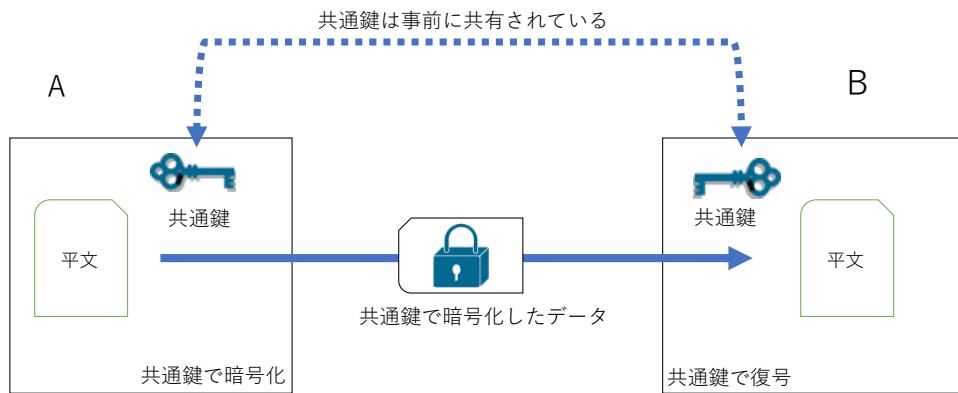


図 2.1: 共通鍵暗号方式での暗号化

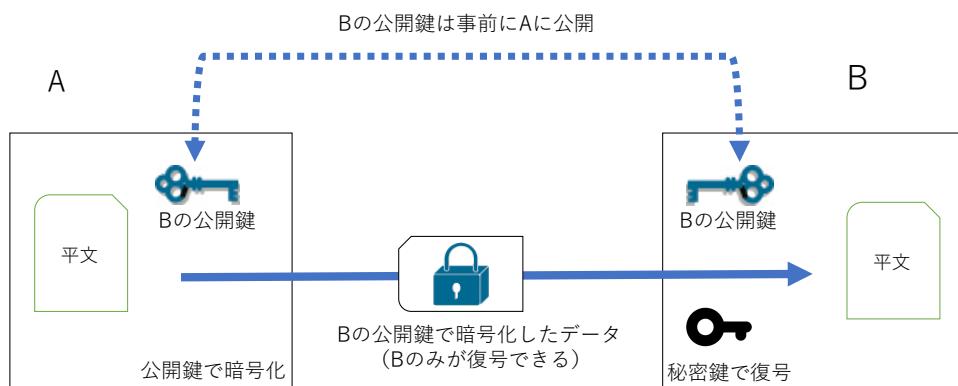


図 2.2: 公開鍵暗号方式での暗号化

2.1.2 公開鍵暗号方式

公開鍵暗号方式は、二種類の鍵である公開鍵と秘密鍵をペアで使用する。この公開鍵と秘密鍵には次に示す性質があり、公開鍵暗号方式はこれらの性質を利用して暗号化や署名を実現している。

- 公開鍵で暗号化した平文は、秘密鍵で復号できる
- 公開鍵で暗号化した平文は、公開鍵では復号できない
- 秘密鍵で暗号化したデータは、公開鍵で復号できる
- 公開鍵から秘密鍵を生成できない。

公開鍵暗号方式での暗号化について上図 2.2 に示す。この図では、鍵ペアを作成した B が、公開鍵を A に公開している。A は、B の公開鍵を使用して平文を暗号化して、B へ送付する。送付された暗号文は B 自身の秘密鍵でのみ復号できる。

代表的な暗号化アルゴリズムに「RSA 暗号」「Diffi-Hellman 鍵共有」等がある。

2.2 トンネリング (Tunneling)

ここでは、トンネリングの概要と 2.5 節に関係する VPN トンネリングプロトコルの説明を行う。

トンネリングとは

トンネリングとは、「物理的、論理的に離れた 2 点間に、閉じられた仮想的な回線を作成すること」である。一般的に、パケットやフレームを他のパケットやフレームの中にカプセル化することで実現される。これにより、プライベートアドレスの隠蔽や、カプセル化後に暗号化をすることでデータの保護を行うことができる。後に説明する VPN においてもトンネリング技術は必要不可欠なものとなっている。ただし、「トンネリング=VPN」であるわけではないという点は重要な点である。

トンネリングの機能を以下に示す。

- データ転送の容易化

トンネリングには、パケットまたはフレーム全体を直接特定の場所に転送する、「宛先を切り替える」機能を持つ。これにより、それらが宛先ネットワークに到着するとそのネットワークを管理している組織のセキュリティやネットワークポリシーによる制御を受けることができる。

- 組み込みセキュリティ機能

トンネリングプロトコルの中では、追加セキュリティ（暗号化、認証など）がプロトコルに組み込まれているものがある。そのようなプロトコル（プロトコル群）の代表例として「IPSec」、「L2TP」や「PPTP」がある。

2.2.1 IPSec

IPSec (IP Security Architecture) は、IP パケットのデータの完全性(Integrity)と機密性(Confidentiality)を保証する機能を持っているプロトコル(プロトコルの集合)であり、ネットワーク層で動作し IP 上(インターネットプロトコル)でしか使用できない。これらの機能を提供するための方法として、IPSec は以下の 3 つの要素を持ち、「AH」、「ESP」、「IKE」などの複数のプロトコルから構成されている。

- 認証(ユーザレベルでなく、パケットレベルの)
データの送信者が間違いない本人であること、及び送信されたデータが受信されたデータと同一であることを検証する動作である。
実現のためのプロトコル: AH、ESP
- 暗号化
適切な鍵を持たない人にはデータを読み取ることができないようにする動作である。
実現のためのプロトコル: ESP
- 鍵管理
送信者と受信者との間でセキュリティ「鍵」の値を一致させたり、取り決めさせたりする動作である。
実現のためのプロトコル: IKE

IPSec を構成するプロトコル

- AH (Authentication Header)
パケットが改ざんされていないかどうか認証を行うが、パケットの暗号化はできない。
RFC2402 [3] で定義されている
- ESP(Encapsulated Security Payload)
パケットが改ざんされていないかどうか認証を行い、パケットのペイロード部の暗号化(DES or 3DES or AES)を行う。
RFC2406 [4] で定義されている。
- IKE(Inernet Key Exchange)
秘密鍵情報の交換を安全に行う。(Diffie-Hellman 鍵共有などを用いる)
RFC2409 [5] で定義されている。

IPSec では、事前に通信を行うホスト同士が認証・暗号化のアルゴリズムや暗号鍵を共有するコネクションを形成する必要がある。この関係を SA (Security Association) という。

IPSec に対応したホストやファイアウォールではデータベース内に複数の SA を保有することができ、ある IPSec のパケットがどの SA に対応しているかは、IPSec のヘッダ中に含まれる SPI(Security Parameters Index) というポインタに指定される。

通信モード

IPSecは通信モードが2つあり、「トンネルモード」と「トランSPORTモード」である。ここでは2つのモードを簡単に説明する。

まず、「トンネルモード」では、それぞれ別のネットワークとして構成されているネットワークで発生する通信パケットを、ゲートウェイでカプセル化して転送する。つまり、元のIPパケットをゲートウェイで「外側」のIPパケットの内部にカプセル化し、元のパケットをペイロード(正味のデータ部分)としてまったく新しいパケットを作成し転送している。その後ゲートウェイのIPに従って経路制御され、通信相手のゲートウェイでカプセル化されたパケットが取り出され、宛先となっているネットワークに配送される。

次に「トランSPORTモード」は、端末間でのIPSec通信を実現するために、転送する前に元のパケットのペイロードだけを暗号化する。このとき、IPヘッダは暗号化されず、そのまま経路制御のために利用される。

2.2.2 PPTP

PPTP (Point-to-Point Tunneling Protocol) [6] はデータリンク層で動作するトンネリングプロトコルの一つである。リモートクライアントからプライベートな企業のサーバーにデータを安全に伝送することをようとするための手段として設計された。サーバ/クライアントモデルで、データリンク層でカプセル化を行う。最も古いVPNプロトコルの一つであるが設定が最も容易で計算速度が速いため、古く遅いデバイスで使用される。深刻なセキュリティ上の脆弱性が存在するため、2020年現在使用することは勧められていない。

PPTPの通信手順

以下にPPTPの通信手順を簡単に説明する。

1. 制御用コネクションの確立(PPTPトンネルのための準備)

トンネルの構築、維持、切断処理に関する制御情報をやり取りするために、トンネル以外に制御用情報の交換のためのコネクションを確立することである。制御コネクションにはTCPを利用している。

2. PPPセッションを利用してPPTPトンネルを構築

ここでは、PPP (Point-to-Point Protocol) [7] が本来持っている仕組みを利用して、PPTPトンネルを構築するための認証及び暗号化の方法を選んで、実際に認証と鍵交換を行う。

2.2.3 L2TP

L2TP (Layer 2 Tunneling Protocol) [8] はL2F(Layer 2 Forwarding Protocol)[9]とPPTPのアップグレードとして設計された。VPNに利用することができるが、L2TPそのものは暗号化の仕組みを持っていないため、IPSecの暗号化機能と組み合わせてVPNの暗号化通信を実現している。L2TPはIPSecとは異なり、実装次第でトンネルするネットワークが必

ず IP ネットワークである必要がない。そして、重要な点として L2TP ではカプセル化されたデータは UDP のデータとして (L2TP のメッセージとして) IP ネットワーク上を配送される。

L2TP の通信手順

L2TP では以下のような手順で通信経路の確立を行う。

1. 初めに制御コネクションを確立してトンネルを構築する。
2. その後同じトンネルを使ってユーザセッションを確立する。
3. 最後に PPP によるリンクを確立する。

2.3 Telnet

Telnet [10] は、ネットワークに接続された機器を遠隔操作するために使用するアプリケーション層の技術である。サーバ・クライアント方式で提供され、Telnet サーバが操作される側、クライアントが操作する側で動作する。Telnet を使うことでオフィスのデスクにいながら、マシンルームにあるサーバ、ルータ等の機器をパソコン上で操作できる。サーバには telnet クライアント、ルータなどの機器には telnet サーバーのサービスが有効であることが前提である。基本的にはポート番号 23 を使用する。

2.3.1 Telnet の仕組み、使用法

Telnet の接続までの流れは以下の図 2.4 のようになっている。クライアントからの Telnet はコマンドプロンプトや Terminal から、「telnet 150.65.136.94」というように telnet コマンドと接続したいサーバの IP アドレスを入力するか、Windows では Tera Term 等で IP アドレスと入力して Telnet 接続を行う。TCP によるコネクション確立後クライアントのコマンドプロンプトで Telnet サーバから応答画面が表示される。Telnet で遠隔操作を行うためには対象の機器にログインする必要があるため、最初の応答画面ではパスワード要求がされる。

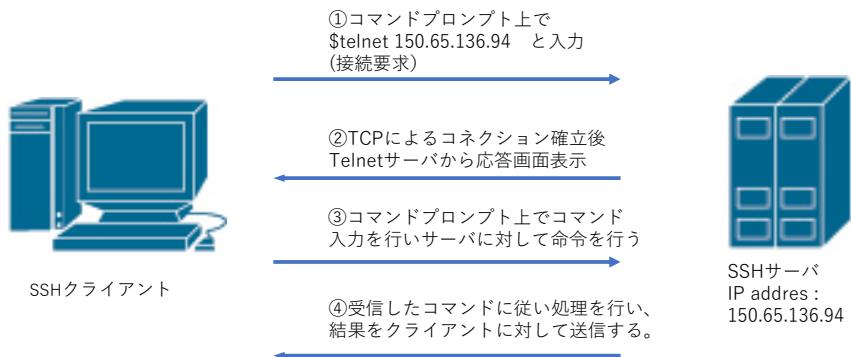


図 2.3: Telnet 接続フロー

問題点として、認証も含めすべての通信を暗号化せずに平文のまま送信するため、パスワードを盗むのは比較的容易である。同様の機能を持ち、情報を暗号して送信することができる SSH が存在し、セキュリティの観点から Telnet よりも SSH が推奨されている。

2.4 SSH

上記の Telnet は、遠隔操作するサーバーの認証情報を含め、通信を暗号化せずに平文のまま通信を行う。その結果、通信内容を盗聴されると認証情報や通信内容が簡単に盗まれてしまう危険性がある。この問題を解決してくれるのが、Telnet と同様な機能を持ちつつ通信内容を暗号化してくれる「SSH」(Secure SHell) [11] である。SSH には「SSHv1」と「SSHv2」の二種類のバージョンが存在し、本論文では安全性の高い「SSHv2」のみの解説を行う。

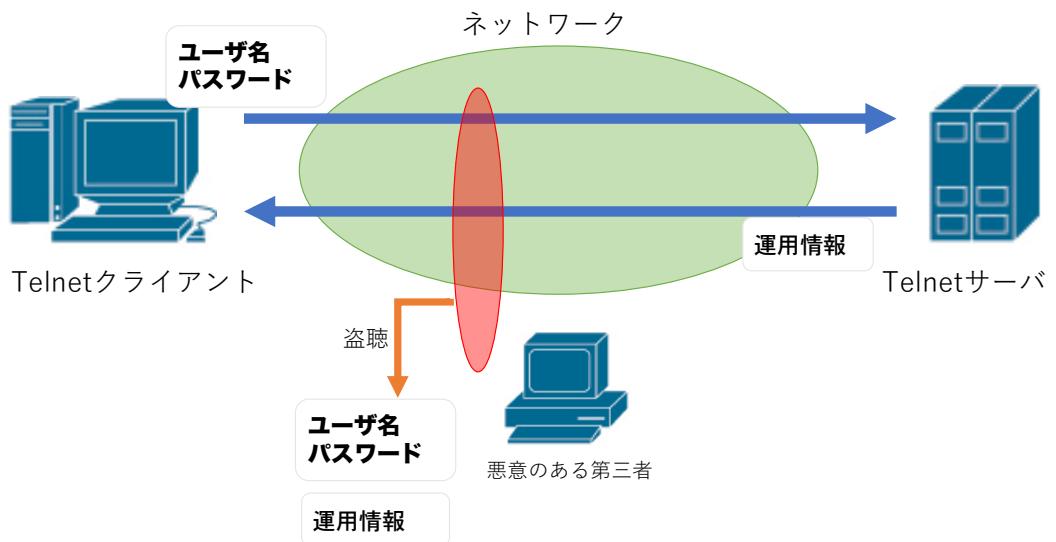


図 2.4: telnet 接続による脅威

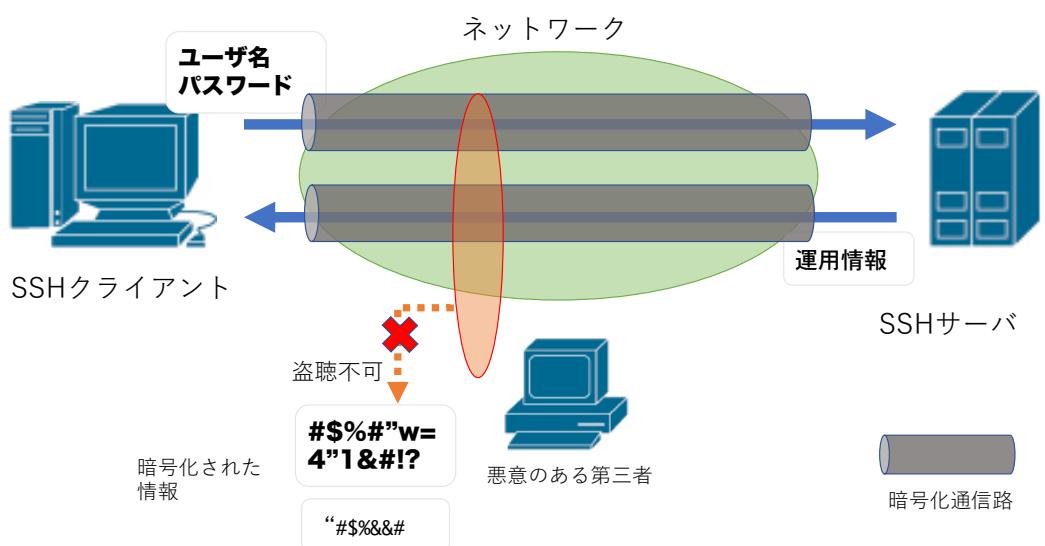


図 2.5: SSH 接続によるセキュアな運用管理

SSH の技術を用いるために、「OpenSSH」というソフトウェアが一般的に使用される。認証方式として大きく 2 つあり、「パスワード認証」と「公開鍵認証」が使われる。「パスワード認証」は、ログイン時に利用するアカウント情報をそのまま異利用し、ID とパスワードが一致すれば認証を行う。「公開鍵認証」は、クライアントが公開鍵と秘密鍵を生成し、クライアント側が持つ秘密鍵が、サーバー側が持つ公開鍵に対応するものであるかどうかで認証する。クライアント側が持つ秘密鍵はネットワーク上に送信されることはなく、サーバ側が持つ公開鍵から秘密鍵を推測されないため、パスワード認証よりも安全な認証を行える。SSH の公開鍵とユーザ ID が “~/.ssh/known_host” ファイルに登録される。

2.4.1 SSH の機能

1. セキュアリモートログイン

通常、Secure Shell (SSH) と呼ばれる機能のこと。セキュアリモートログインを使用すると、インターネット経由でも安全に、運用端末から SSH サーバへログインできる。また、通信内容を他者に見られないとため、安全な運用管理を実現できる。

使用コマンド例 \$ ssh pc15@150.65.136.94

2. セキュアコピー (scp)

セキュアコピーを使用すると、SSH サーバからファイル転送を受け取ることできる。また、通信内容を他者に見られたり、改ざんされたりすることがないため、安全な運用管理を実現できる。

使用コマンド例 \$ scp -r pc15@150.65.136.94:/home/pc15/画像 /Users/shuto/desktop
これにより、ssh サーバから、画像ディレクトリをファイル転送を受け取れた。

3. セキュアFTP(sftp)

「SSH」で暗号化された通信路を使って FTP を使用している。セキュアFTP を使用すると、SSH サーバーにファイルを転送することができる。セキュアコピーと同様に、通信内容の盗聴や、改ざんを防ぐことができる上、セキュアコピーではできないファイルの名前変更や削除などが可能である。

2.4.2 SSH 接続までの流れ

まず初めに暗号化通信路の確立までの流れを下図 2.6 のように示す。

(a) バージョンと各種暗号方式の交換

初めに、サーバとクライアントの間で使用する SSH バージョン情報を交換し、SSHv1、SSHv2 のどちらを利用するかを決定する。その後、使用できる鍵交換方式、希望する公開鍵暗号方式、共通鍵暗号方式、メッセージ認証コード、のアルゴリズムの各リストを交換する。

(b) ホスト認証と暗号化通信路の確立

各 SSH サーバは、それぞれ異なるホスト鍵ペア（ホスト公開鍵とホスト秘密鍵）を保持している。ホスト鍵ペアはインストール時に生成される。クライアントは、サー

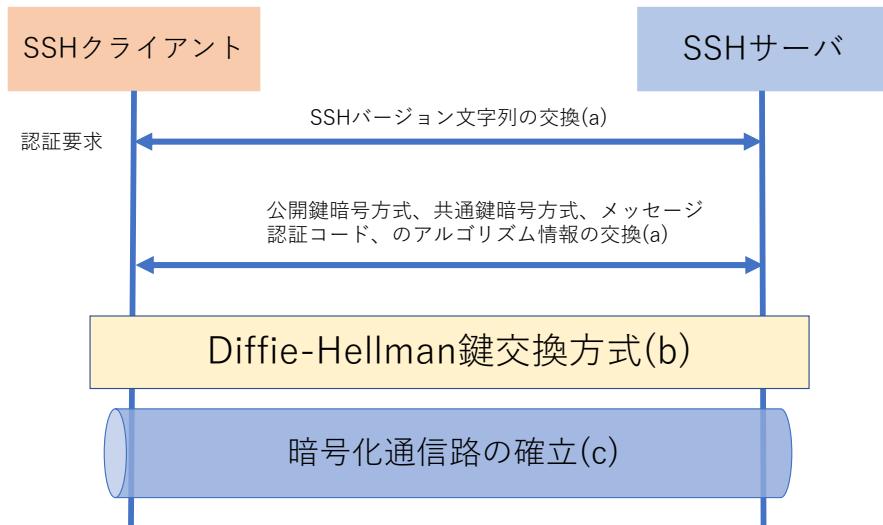


図 2.6: SSH 接続確立までのフロー

バの認証、識別のためにこれらの鍵を使用する。サーバ及びクライアントは、交換した共通鍵暗号方式やメッセージ認証コードのリストから、使用するアルゴリズムを決定する。その後、Diffie-Hellman 鍵交換方式で、暗号化通信路に使用する共通鍵を交換する。共通鍵の交換中にサーバの持つホスト公開鍵をクライアントで保持しているホスト公開鍵のデータベースと照合して、サーバの認証も行う。ここで、Diffie-Hellman 鍵交換方式は、交換する鍵を直接送ることなく、両者で鍵を共有できるアルゴリズムである。

(c) ユーザ認証

ホスト認証後、暗号化通信路が確立されると、公開鍵暗号方式またはローカルパスワードによるユーザ認証を行う。

(1) 公開鍵暗号方式によるユーザ認証

接続先のサーバにはあらかじめユーザの公開鍵を登録しておく。クライアントでは、登録されているユーザ公開鍵に対応した、ユーザが所持している秘密鍵を使用して認証する。SSHv2 では、「電子署名」という方法を使用する。

まず、クライアントでは、ユーザ名、ユーザの公開鍵、ユーザの公開鍵アルゴリズムを記述した認証要求メッセージを作成する。そして、作成した認証要求メッセージに対して、ユーザの秘密鍵を使用して電子署名を作成する。最後にサーバーに対して認証要求メッセージに最後に、サーバに対して、認証要求メッセージに電子署名をつけたものを送付する。

サーバでは、送付された認証要求メッセージから、ユーザ名とユーザ公開鍵を取り出し、登録済みのユーザとユーザの公開鍵であることを確認する。また、登録されているユーザの公開鍵を使用して、送付された電子署名を審査し、正しいユーザの電子署名であることを確認できることを確認すると、ユーザの認証成功となる。

(2) ローカルパスワードによるユーザ認証

Telnet と同様に、サーバでローカルに設定されたパスワードを使用してユーザ認証を行う。しかしパスワードは暗号化された通信路を経由するため、第三者には見えない

(d) ログイン後

ユーザ認証に成功すると、セッションが確立し、ユーザはログインする。ここで通常はターミナルのセッションが開始される。

2.5 VPN

VPN とは

VPN とは、「Virtual Private Network」の略であり、「仮想専用線」や「仮想閉域網」と訳され、通信事業者のネットワークやインターネットなどの公衆ネットワーク上で作られる、仮想的な専用ネットワークの総称である。VPN を利用することで、公衆回線（インターネット）での脅威を防ぎ、安全なリモートアクセスを実現することができる。VPN の実現のための主な方法として、ここでは「IPSec-VPN」と「SSL-VPN」を挙げる。まず初めに下表 2.1 に 2 つの違いを簡単にまとめ、その後各々の説明を行う。

表 2.1: IPSec-VPN と SSL-VPN の違い

	IPSec-VPN	SSL-VPN
リモートアクセス端末への専用ソフトインストール/環境設定	必要 環境設定も複雑 (専用ソフトは、IPSec-VPN装置と同一メーカー製品が原則)	不要 専用ソフトが必要な場合は自動インストール、自動環境設定
リモートアクセス端末機器	△ 専用ソフトが対応している装置 (パソコンが中心)	○ パソコン、 携帯電話 (WEBブラウザ使用)
コンテンツやサーバに対するアクセス制御	△ 難しい	○ 容易
初期導入コスト	○ 低い	△ 高い
運用管理コスト	△ 高い	○ 低い
既存ネットワークへの適用性	△ NAT (アドレス変換)、ファイアウォール越えなどの考慮が必要	○ シームレスに導入可能
性能 (処理速度・アクセス速度)	○ SSL-VPNより高速	△ IPSec-VPNより低速

2.5.1 IPSec-VPN

IP-VPN とは、2.2 節で説明した IPSec の技術を利用して VPN を構成する方法である。IPSec-VPN では、VPN ゲートウェイ装置との間に VPN トンネルを作るため、リモートア

クセス端末に専用のソフトをインストールする必要がある。また、暗号化や認証のための設定などの環境設定を事前に行う必要があり、IPSec-VPN は、複数のオフィス間で VPN を張る等、固定的なエンドポイント同士での VPN 接続に向いている。

モバイルデバイスと社内システムを VPN で接続するなどには、より手軽に利用できる SSL-VPN を使用するのが一般的である。

2.5.2 SSL-VPN

SSL-VPN とは、暗号化に SSL 技術を使用した VPN である。IPSec-VPN はクライアント PC に必ず VPN のクライアント用ソフトウェアをインストールする必要があるのに対して、SSL-VPN の場合は Web ブラウザさえあれば通信可能である。

SSL-VPN には、「リバースプロキシ」、「ポートフォワーディング」、「L2 フォワーディング」の 3 つの方式がある。まずははじめに 3 つの方式の比較を簡単に下表 2.2 に示した後、それぞれを簡潔に説明する。

表 2.2: SSL-VPN の実装方式の比較

	リバースプロキシ方式	ポートフォワーディング方式	L2フォワーディング方式
リモートアクセス端末側構成要素	WEBブラウザ	WEBブラウザ+モジュール (WEBからダウンロード、自動インストール)	WEBブラウザ+モジュール (WEBからダウンロード、自動インストール)
使用可能アプリケーション	△ WEBアプリケーション	○ 通信中ポート番号が変わるもののは使用できない場合あり	◎ ほとんどのアプリケーションで使用可能
リモートアクセス端末機器	○ WEBブラウザが動く端末	△ モジュールの仕様によって制限 利用時に管理者権限が必要	△ モジュールの仕様によって制限
用途	出張先の端末などから簡単に使いたい。 仕様アプリケーションは WEB メールや WEB 型グループウェアなど WEB ページ中心	クライアント端末の OS が様々である。 ある程度の種類のアプリケーションを使いたい	アプリケーションを制限なく使いたい。 使用されるクライアント端末の種類は限られている。

SSL-VPN(リバースプロキシ方式)

「リバースプロキシ」は、通常のクライアント側（利用者側）の「プロキシ」ではなく、サーバー側（アクセスされる側）に位置するものである。簡単な構成を下図 2.7 に示す。

クライアントからは Web サーバにアクセスしているように見えるが、実際は SSL-VPN Gateway（リバースプロキシ）が仲介者として Client の Request に沿って各ローカルの Web サーバにアクセスしている。

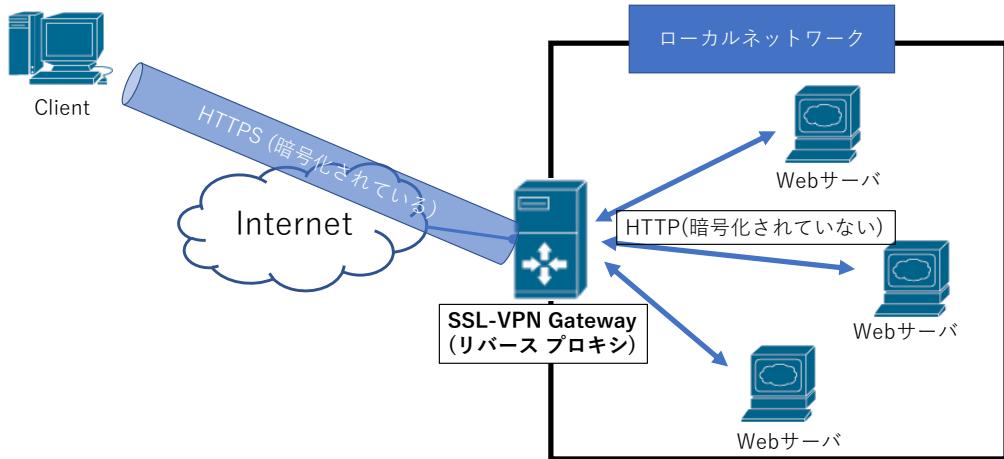


図 2.7: リバースプロキシの図

これにより、Web サーバに外部からの直接的なアクセスができなくなり、改ざんや不正侵入などのリスクを減らすことができる上に SSL-VPN Gateway にファイアウォールや認証機能を追加することでセキュリティの強化を行うことができる。また、SSL-VPN Gateway だけに SSL 証明書を適用すればよくなる。

リバースプロキシ方式は、Web ブラウザで動作しないアプリケーションは使用できないという問題があった。そこでリモートアクセスしてくるクライアントに Java や ActiveX で作成されたモジュールを追加させた上で SSL-VPN 通信を行う方法が考えられた。それが、「ポートフォワーディング」と「L2 フォワーディング」である。

SSL-VPN(ポートフォワーディング方式)

一般的に上記のリバースプロキシ方式では、SSL-VPN Gateway にファイアウォールを設置し、外部から利用できるアプリケーションを制御（Web サーバは閲覧できるが、Telnet はできないなど）している。その制御には、通信されるデータに含まれる、ポート番号と呼ばれるアプリケーションの種類を示す情報を使用している。

ポートフォワーディングとは、ファイアウォールを通過できないアプリケーションのデータのポート番号を、通過できるアプリケーションのポート番号に変換することにより、ローカルネットワークとグローバルネットワーク（インターネット）との通信を可能にする機能である。ポートフォワーディング方式では、この機能を用いて任意のアプリケーションの通信を上図 2.8 のように HTTPS のポート番号に変換し、ファイアウォールを通過させることで SSL-VPN を実現する。

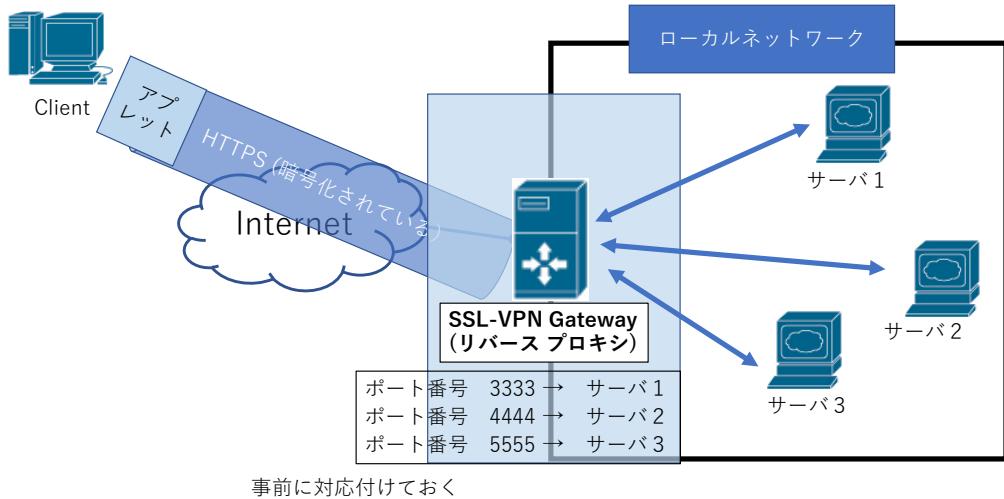


図 2.8: ポートフォワーディングの図

SSL-VPN(L2 フォワーディング方式)

L2 フォワーディング方式ではクライアント PC に SSL-VPN クライアントソフトをインストールする。

L2 フォワーディングでは、アプリケーションのデータを HTTP パケットでカプセル化して SSL 通信を行う。ポートフォワーディング方式のように、SSL-VPN Gateway で通信するサーバの IP アドレス/ポート番号を事前に定義する必要がないため、幅広いアプリケーションをサポートすることが可能である。

専用のソフトウェアをインストールすると、クライアントに SSL-VPN 用の仮想 NIC が追加され、仮想 NIC 経由の通信がすべて SSL で暗号化される。仮想 NIC の IP アドレスは、通常 SSL-VPN 接続とともに自動設定される。そして、クライアントは仮想 NIC によって社内ネットワークに直接接続されているのと同等に扱うことができるようになる。

企業において SSL-VPN を用いてテレワークなどのために社内ネットワークに遠隔アクセスする場合、ログアウト後にそのクライアント PC (テレワーク利用端末) にデータが残らないよう自動的にデータが削除されるように実装させるのが一般的である。

参考として、Cisco の VPN ゲートウェイ装置で SSL-VPN を行う場合、「リバースプロキシ方式」、「L2 ポートフォワーディング方式」が利用できる。「ポートフォワーディング方式」は利用できない。

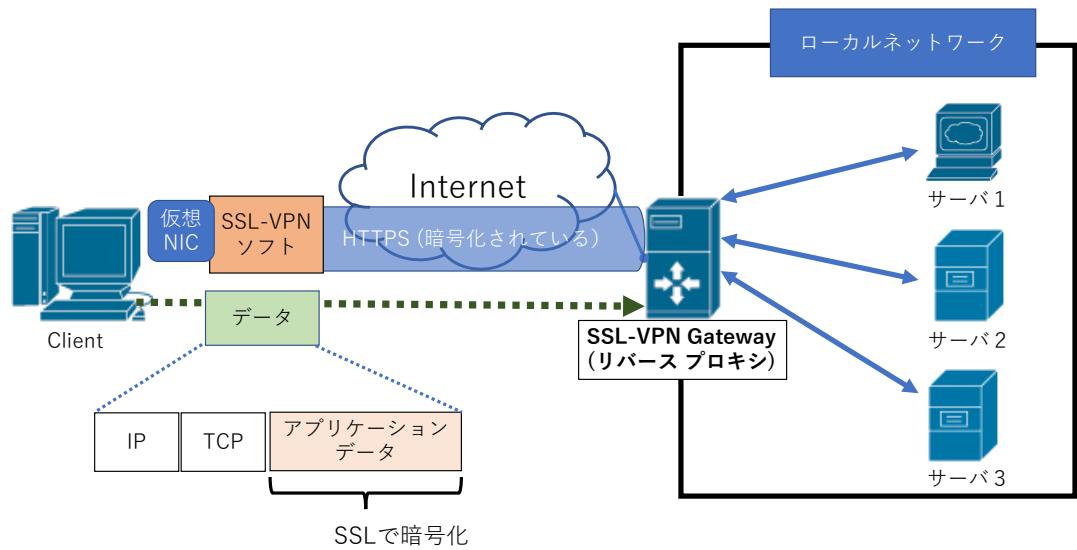


図 2.9: L2 フォワーディングの図

第3章 認証技術

ここでは認証技術として代表的な技術である、「LDAP」、「Kerberos」、「RADIUS」、「Active Directory」の解説を行う。

3.1 LDAP

概要

LDAP(Lightweight Directory Access Protocol) [12] は、Active Directory のようなディレクトリサービスにアクセスするためのプロトコルである。LDAP 自体はプロトコルであり、サービスやシステムを指すものではない。LDAP 認証は、ユーザの名前とパスワードの組み合わせの検証にのみ使用される。そのため、LDAP を利用して認証を行うようにする前に、ユーザ情報はデータベースに存在しなければならない。

代表的な LDAP を利用できるソフトウェアに、「Open LDAP」、「Active Directory」が存在する。

ディレクトリサービス

ディレクトリサービスとは、ディレクトリと呼ばれるデータベースから、ユーザ名やマシン名などのキーを元にデータを検索、参照するためのサービスである。一般的にデータベースと呼ばれる、SQL 言語等を用いて扱う「RDB (Relational Data Base)」では、データ間の関係性を利用して、データの参照、挿入、更新、削除、といった操作を行うため、それらは少し異なるものである。グローバルでサービスを提供する場合には、分散型ディレクトリサービスが用いられ、DNS(Domain Name Service) が分散型ディレクトリサービスとして有名である。

LDAP、ディレクトリサービスの特徴

- 読み取りが高速
- 分散型の情報格納モデル
- 高度な検索機能をもつ

LDAP の特性としては、「情報の参照、検索」に特化している。このようになる理由としては、ディレクトリサービスとして利用されるものは、一般的なデータベースのように読み取りと書き込みが同じ頻度で発生することではなく、大規模システムではユーザ情報の

利用は参照検索が最も頻繁に起こるため、それらの操作に対する高い性能が必須であるため、このような特性となっている。

LDAP を用いたデータの集中管理の様子を下図 3.1 に示す。

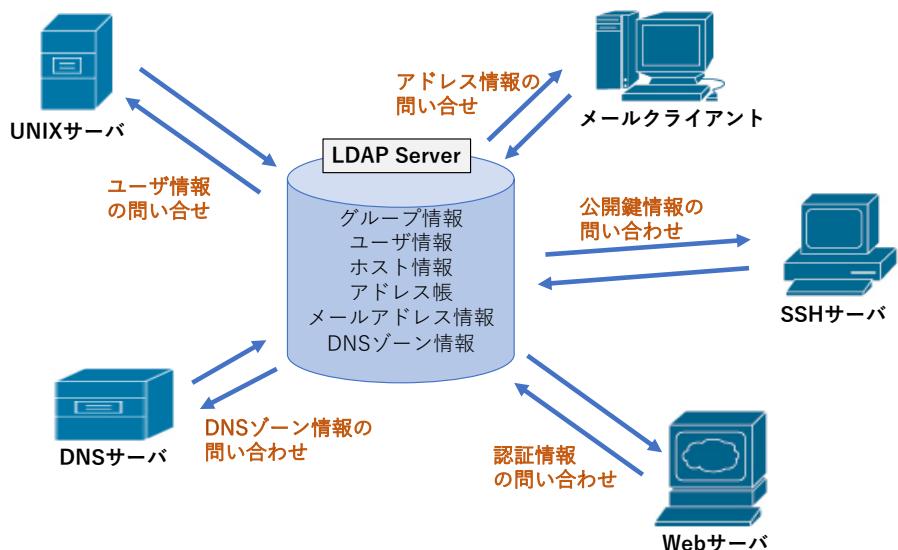


図 3.1: LDAP を用いてデータを集中管理の図

LDAP できること

1. リソースの一元管理
多数のクライアントがある場合、1台1台にID・パスワード情報を入ることなく、LDAPサーバー1台だけ登録すれば、どのクライアントからも同じID・パスワードでログインできるようになり、さらに環境もログイン時に取得できるようになる。
2. リソースのアクセス制御
特定のIPアドレスからであれば、読みと書き可能であるが、それ以外からは読みしかできない。
3. 各種サービスとの連携
多くのアプリケーション(Open Source Software)と連携することができる。初期のユーザ情報作成をLDAPデータベースから行い、認証をLDAPサーバに委譲することができる。これの発展形として、一つのサーバで認証すれば、他のサーバでは認証無しでログインできる仕組みである、シングルサインオン(SSO)を実現できる。

3.2 Kerberos

Kerberos とは

Kerberos [13] とは、ネットワーク上でユーザの認証を行う方式の一つであり、サーバとクライアント間の身元確認のために使用される。クライアント/サーバ間の通信を暗号化でき、比較的セキュリティが強固な認証方式となっている。ケルベロス認証では一度ログインすると、「チケット」と呼ばれるものを用いて認証を行えるようになるため、次回のログイン時に ID・パスワードを改めて入力する必要がなくなり、シングルサインオン (SSO) を実現できる。

利用例としては、Active Directory のユーザ認証の際に用いられている。名前はギリシャ神話の地獄の門を守る番犬ケルベロスに由来している。

Kerberos の構成要素

Kerberos の仕組みを解説する前に、用語「KDC、AS、TGS、プリンシパル、realm」の説明を行う。

- KDC (Key Distribution Center)
サーバとユーザに関する信頼関係の情報を一括管理する中央データベース。これを LDAP サーバにすることもできる。
- AS (Authentication Server)
認証サーバで、ユーザからの認証を受け付けるサーバ。
- TGS (Ticket Granting Server)
チケット発行サーバ。各サーバを利用するためのチケットを発行するサーバ。
- プリンシパル (principal)
KDC 認証を行うユーザやサーバのこと。
- realm (realm)
同じ KDC の配下にあるシステムをグループとして定義する論理ネットワーク。

これらを構成すると以下の図 3.2 になる。

Kerberos の認証の仕組み

Kerberos 認証では、ユーザが正しいユーザ ID とパスワードを AS (Authentication Server) に送信して認証に成功すると TGS(Ticket Granting server) からチケットと呼ばれるデータを受け取れる。Kerberos 認証ではこのチケットを認証に使用する。サーバはアクセスしてくるユーザがアクセス権を持っているかどうかをユーザ ID とパスワードではなくチケット（クライアント ID、タイムスタンプ、有効期限が記されている）を使用して確認する。認証時にチケットを使用することでアカウント（ユーザ ID、パスワード）の漏洩を防いでいる。全体的な流れを以下図 3.3 に示す。

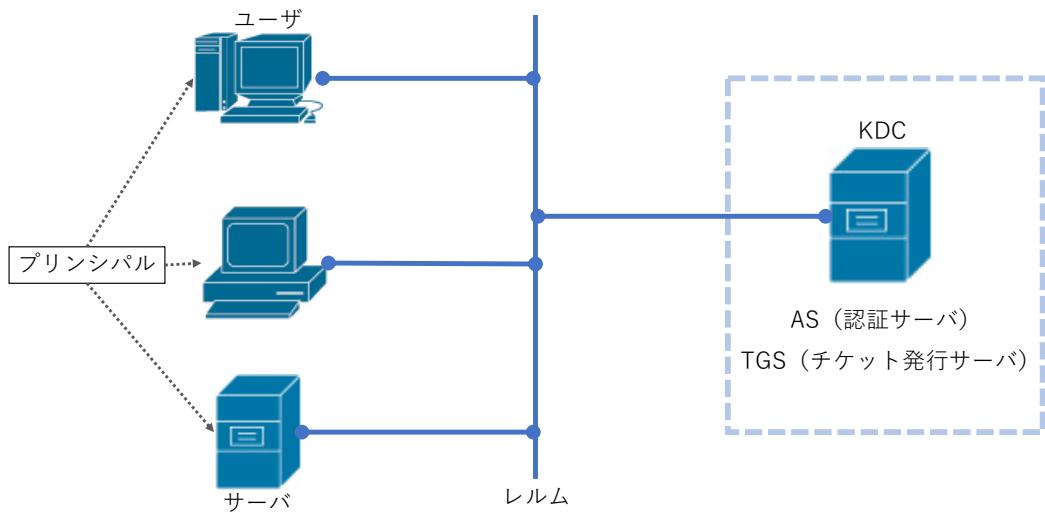


図 3.2: Kerberos 認証の構成要素

Kerberos 認証では、チケットの盗聴によるなりすましを防ぐために、時刻同期の仕組みが用意されている。チケットの中にはタイムスタンプ（送信時刻）が記録されており、チケットを受信したサーバがチケットのタイムスタンプとサーバの持つ時刻と 5 分以上のズレがあると認証に失敗するようになっている。したがって、NTP（Network Time Protocol）を使用して、チケット発行側の時刻とチケット利用側の時刻が同じにする必要がある。

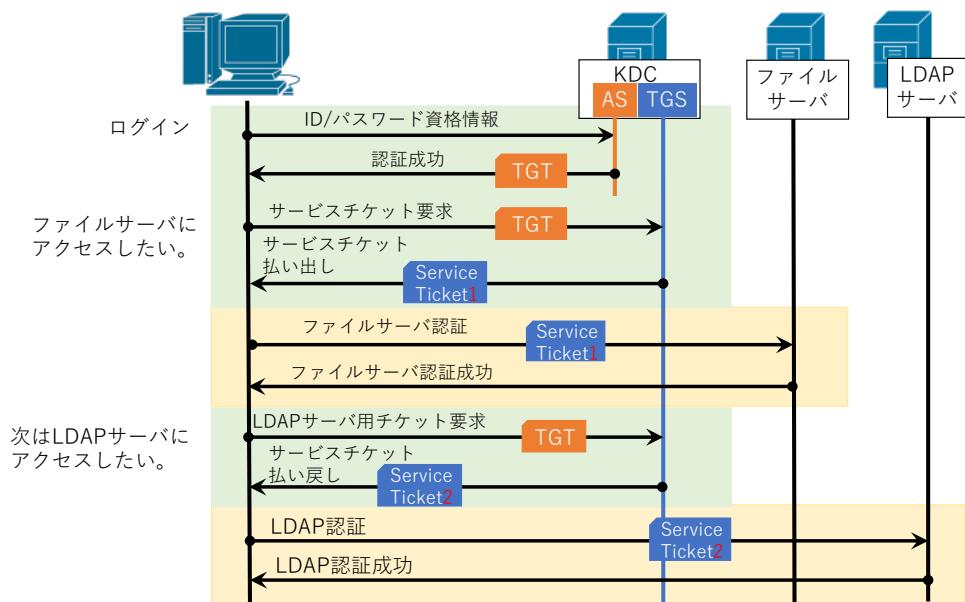


図 3.3: Kerberos 認証流れ

3.3 RADIUS

RADIUS とは

RADIUS(Remote Authentication Dial In User Service) [14] は、ネットワーク上のユーザ認証プロトコルである。RADIUSによる認証システムは、「RADIUS サーバ」、「RADIUS クライアント」、「ユーザ」の3つの要素で構成されている。

インターネットが普及を始めたころ、ユーザがインターネットへのアクセスするには電話回線を使ったダイヤルアップが主流であったため、RADIUS サーバはダイヤルアップサービス用の認証サーバとして開発された。その後 ISP や企業などで利用されるようになり、ネットワークが光回線に置き換わった現在でも、ISP の認証サービスなどでは RADIUS サーバが継続して使用されている。また、近年では、Wi-Fi アクセスポイントでの認証などでも、RADIUS サーバが使われている。

- **RADIUS クライアント**
アクセスしてくるユーザの認証要求を受け付けて RADIUS サーバにその情報を転送する役割。NAS (Network Access Server) とも呼ばれる。
- **RADIUS サーバ**
認証要求に応じて認証を実行してアクセスを許可するかどうかを決定する役割。

RADIUS 認証の流れ

ユーザがネットワークやネットワーク機器を利用したい場合、全体の認証の流れは下図 3.4 のようになっている。

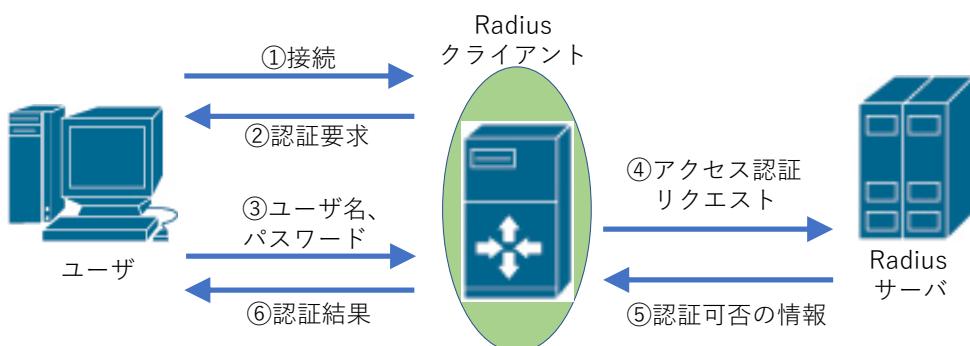


図 3.4: RADIUS 認証の流れ

1. ユーザがネットワークに接続しようとする。

2. RADIUS クライアントは、ユーザに認証を要求する
3. ユーザがユーザ名、パスワードを入力し、認証を行う。
4. RADIUS クライアントは、受け取ったユーザ名、パスワードを使って、RADIUS サーバにアクセス認証の Request を出す。
5. RADIUS サーバは認証処理を行い、RADIUS クライアントに認証可否を伝える。
6. RADIUS クライアントは、ユーザに認証結果を伝える。

3.4 Active Directory

Active Directory とは

マイクロソフトによって開発されたディレクトリサービスシステムで、一般的に WindowsOS で使用される。Active Directory は複数のサービスの総称であり、「Windows システムで認証を行う機能」を持つ。

社内システムにおいて、利用権限によって一部の社員にしかアクセスさせたくないシステムに対して、Active Directory の認証機能を使用して、アクセスの制限を行う。

Active Directory の機能

Active Directory は 5 つのサービスから構成されている。

- Active Directory ドメインサービス (AD DS)
ユーザー やコンピュータの認証や、管理者が情報を安全管理したり、ユーザがファイルやディレクトリなどのリソースを簡単に検索することができる機能である。
一般的に Active Directory というと、このドメインサービスのみを指すことが多い。
- Active Directory ライトウェイトディレクトリサービス (AD LDS)
AD DS の簡易版という立ち位置。AD DS の構成要素のうち「データベースの仕組み」、データの検索記録が行えるようになったもの。認証を行う機能はない。
- Active Directory 証明書サービス (AD CS)
公開鍵 (PKI) を構築するための、証明書の作成と管理を行う証明機関 (Certification Authority:CA) を作成するサービス。
- Active Directory Rights Management Services (AD RMS)
ドキュメントの権限管理やコンテンツ保護など、不正使用から情報を保護するための機能。AD RMS を利用することで、メールやドキュメントの保護が可能になり、保護されたデータは暗号化される。
- Active Directory フェデレーションサービス (AD FS)
複数の Web アプリケーション間の認証や、異なる組織間での認証など、組織の違いを超えて認証の仕組みを連携する機能。

第4章 研究のためのネットワーク構成

本研究では、L2スイッチ1台とCentOS8.0のインストールされたサーバ3台を用いて1台をGateway Server、残りの2台をHost Serverとしている。以下図4.1のような隔離ネットワークを構成し、そのネットワーク内でソフトウェアを試用する。ここで隔離ネットワークとは、内部のHost Server2台はGatewayServerを経由しないと外部からは接続できないようになっているネットワークと定義する。L2スイッチの設定は、PC7とGateway間、PC8とGateway間のそれぞれの間は”ping”コマンドが通るようにし、PC7とPC8間は”ping”コマンドが通らないように設定した。

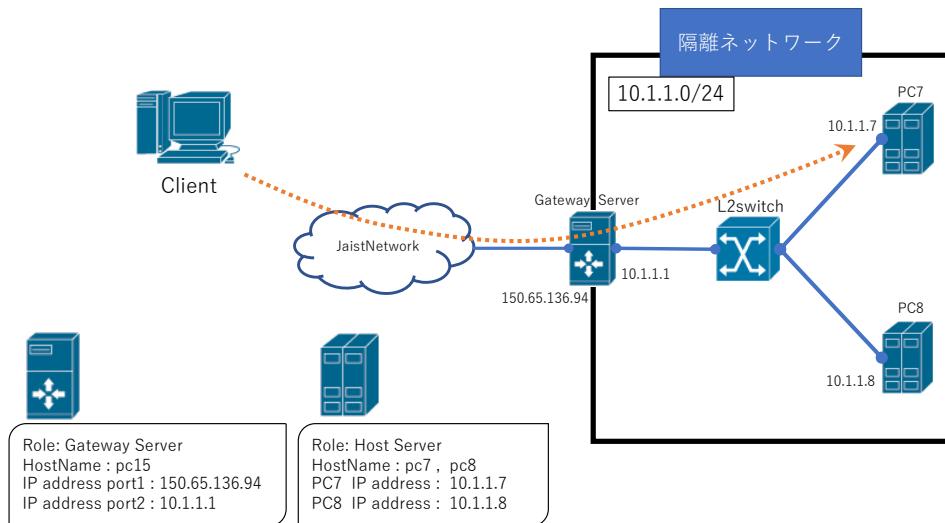


図4.1: ネットワーク構成図

使用機器の説明

ここでは、HostServer、GatewayServerとL2スイッチの使用機器の説明を行う。

まず、HostServer2台とGatewayServer1台はそれぞれ同じ機器であり、計3台を使用している。1台の詳細を下表4.1に記す。

表 4.1: サーバーの詳細

Serverのメーカー	Supermicro
Serverのモデル	SYS-5018D-FN4T
CPU	Intel(R) Xeon(R) CPU D-1541 @ 2.10GHz コア 8コ, スレッド 16コ, 物理CPU 1コ
マザーボード	メーカー : Supermicro 型番 : X10SDV-8C-TLN4F
メモリ	128GB
ストレージ	モデル : ATA SAMSUNG MZ7LM240 (scsi) ハードディスク /dev/sda: 240GB セクタサイズ (論理/物理): 512B/512B ×2
ネットワーク	メーカー : Intel 2x Gigabit Ethernet LAN ports 2x 10GBase-T ports 1x Dedicated IPMI LAN port

そして、次に L2 スイッチのスペックの詳細を下表にまとめる。

表 4.2: L2 スイッチの詳細

メーカー	FXC株式会社
製品型番	FXC5210
データ転送速度	10/100/100Mbps(CSMA/CD)
イーサネットポート 10BASE-T / 100BASE-TX /1000BASE-T	8ポート 2SFP(Small Form Factor Pluggable)
総スループット	14.8Mpps(64byte)
総帯域幅	20Gbps
MACアドレス登録数	8,000個

本研究でのアプリケーション規模の定義

そして、本研究では、試用するソフトウェアを導入規模で小規模と大規模というように分類し、各種機能で細分化して評価をしていくが、ここで小規模アプリケーションと、大規模アプリケーションを以下のように定義する。

- 小規模アプリケーション

上図4.1において、「GatewayServer」だけにインストールするもの、「Client」だけにインストールするものと定義する。

- 大規模アプリケーション

上図4.1において、「GatewayServer」、「Client」、「HostServer」のそれぞれにインストールを行い、各種設定をおこなう必要があるソフトウェアと定義する。

次の章で、小規模ソフトウェアの試用を行う。

第5章 小規模ソフトウェア

インストール先がゲートウェイサーバーだけのものや、隔離ネットワークにアクセスするクライアントだけのものを小規模のアプリケーションと分類して、評価を行う。まず最初に、評価表を表 5.1 に示す。

表 5.1: 小規模ソフトウェア比較表

名前	概要	ログ能力	鍵交換	webUI	適用先	適用難易度	
sshportal	動的にユーザとホストを構成するBastionサーバーツール	あり	RSA	なし	Gateway Server	少し難しい	導入や使用法などがあまり詳細に記されていない。
sshuttle	擬似的な簡易VPN	あり	※VPN	なし	クライアント	易しい	専用のサイトが用意されている上に、導入手順が詳細に記されている。
sshpiper	プロキシーのようなソフトウェア	あり	RSA	あり	クライアント	すこし難しい	導入や使用法などがあまり詳細に記されていない。

5.1 sshuttle

概要

“sshuttle(<https://github.com/sshuttle/sshuttle>)”は、簡易 VPN ツールである。リモートアクセスユーザ (Client) のみにインストールすれば使用できる。本論文では、図 4.1 の Client にインストールを行なった。

インストール方法

今回、クライアント pc には MacBook を用いたため、homebrew を使ってインストールを行なった。他の OS の場合のインストール方法もいくつかここに記す。

- Ubuntu

```
$ apt-get install sshuttle
```

- MacOS

```
$ brew install sshuttle
```

- Centos

```
$ git clone https://github.com/sshuttle/sshuttle.git
```

```
$ cd sshuttle
```

```
$ sudo ./setup.py install
```

Clientのみにインストールすればよいため、とても容易に使用することができる。

使用方法

今回、図4.1でClientと隔離ネットワークとVPNを形成したい時を想定する。

”\$ sshuttle -r pc15@15.65.136.94 10.1.1.0/24”のコマンドで、図5.1のようにVPNを形成できた。一度VPNを形成すると、多段sshをする必要などなく、自由に隔離ホストにアクセスできるようになる。

例えば、一度sshuttleでVPNを構築すると、Clientの端末上で

\$ ssh pc8@10.1.1.8 のコマンドで隔離サーバであるPC8に直接ssh接続することができる。

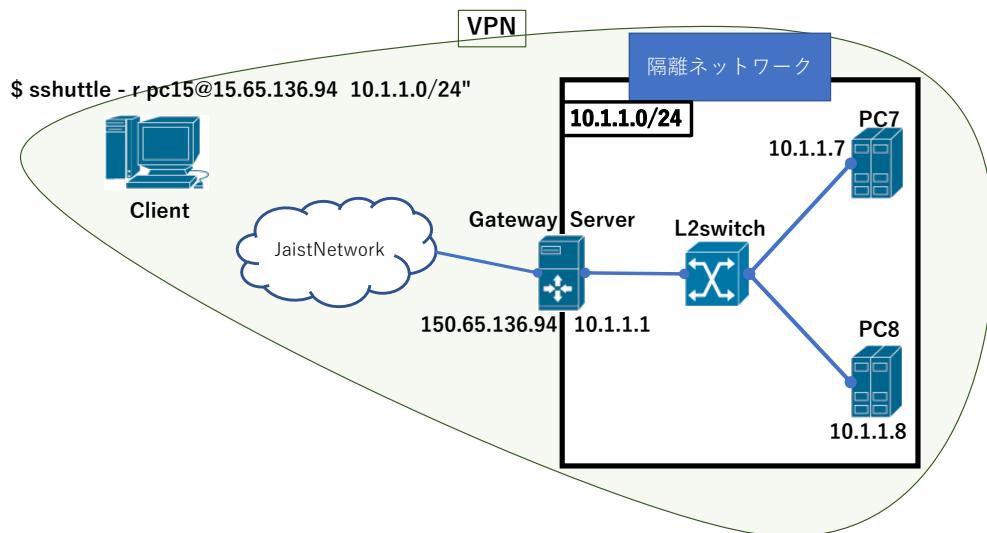


図5.1: sshuttleのVPN構築イメージ図

評価

sshuttleを使用するメリットは、「安全性」と「簡易化」が可能になることである。

一般的に、図4.1のような隔離ネットワーク内のホストにアクセスするためには、多段 ssh や Tunneling を行う必要がある。多段 ssh というのは今回の場合、Client がまず Gateway Server に ssh を行い、その後 Gateway Server のコマンドラインから、隔離ホストに ssh 接続するというよう複数回 ssh を行うことである。この場合、一度 ssh を切ってしまうと、再び多段 ssh する必要である上に、各プロセスごとに多段 ssh をする必要がある。しかし、sshuttle を使用することで、一度 VPN を構築すると自由に隔離ホストにアクセスすることができるため、とても容易に隔離ホストに接続できる。

また、安全性の面では一般的に多段 ssh を行う場合は、Gateway Server を経由するため、外部ネットワークと接続されている Gateway Server に Client の ssh key を保存しないといけない。しかし、sshuttle で VPN を構築することで Gateway Server を経由せずに隔離ホストと接続するため、Client と隔離ホスト間で鍵共有するだけでよいため、インターネット等と接続されている Gateway に鍵を保存するより安全である。

5.2 sshportal

概要

“sshportal(<https://github.com/moul/sshportal>)”とは、透過的な SSH 要塞サーバーにするソフトウェアである。Gateway Server のみにインストールすれば使用することができる。sshportal を使用することで、管理者は Gateway Server にアクセスし隔離ホストにログインでき、ユーザーを動的に管理することができる。これによって複数のユーザーを複数のホストに簡単に割り当てる様になる。Gateway Server のみが両側に関する情報を知っているため、ユーザはホストを知る必要がなく、アクセスする必要があるすべてのものに自動的に接続される。

インストール方法

sshportal は Docker を用いることで容易にインストールができる。また、今回使用した Gateway Server の OS は CentOS8.0 であるため Docker は使用できない。代わりに”Podman”が存在しているため、そのコマンドもここに記す。Podman の使用法はほとんど Docker と違いはない。

- Docker
docker pull moul/sshportal
- Podman
podman pull moul/sshportal

使用方法

ここでは「podman」使用時のコマンドを示す。

管理者の場合

- バックグラウンドサーバーを開始する

```
podman run -p 2222:2222 -d --name=sshportal -v "$(pwd):$(pwd)" -w "$(pwd)" mou1/ssh-  
portal:v1.10.0
```

- ログを表示させる

```
podman logs -f sshportal
```

- 管理者 (admin) としてログイン

```
# ssh localhost -p 2222 -l admin
```

その後 config >に切り替わる。ここで動的にユーザー登録を行う。

もし、サーバーにアクセスしたいユーザーがいるときの使用法

- 最初に admin ホストを作成する

```
config>host create user1@10.1.1.8
```

- サーバーに鍵を追加する

```
$ ssh user1@10.1.1.8 "$(ssh localhost -p 2222 -l admin key setup default)"
```

ユーザの招待

- 例： config>user invite bob@example.com

これによってユーザーを招待している。このコマンドでは、リモートサーバーにユーザーを作成するのではなく、sshportal.db というデータベースにアカウントを作成する。

ユーザーの場合

- \$ ssh localhost -p 2222 -l 10.1.1.8

これにより、user1 は Gateway から pc8 に接続することができる。

評価

sshportal はユーザの管理が簡単にできるように設計されており、新規ユーザの追加等を管理者が動的に行うことができる。インストール先は Gateway Server のみで良いが、前述の「sshuttle」のように Github とは別に専用のサイトは用意されておらず、Github にも詳細には仕組みやインストール方法が記されていないため、導入や使用までにすこし試行錯誤が必要である。

他のデメリットとしては、ユーザーの管理を自動ではなく、動的に管理者が行う必要があるため、ユーザーが大人数になるほど管理が大変になってしまう。

第6章 大規模ソフトウェア

インストール先が Gateway Server だけでなく、隔離ネットワーク内のホスト全てに Role 別に設定を行う必要のあるものを大規模アプリケーションと分類して、評価を行う。まず最初に、評価表を下表 6.1 に示す。

表 6.1: 大規模ソフトウェア比較表

名前	概要	ログ能力	key exchange	webUI	適用先	適用難易度	
Aker	FreeIPAを利用した Bastionサーバーツール	あり	Kerberos チケット	なし	Gateway Server, HostServer, クライアント	難しい	python2とpython3 の依存関係の問題あり。 インストールまでの説明が少ない。
teleport	リモートアクセスするためのセキュリティゲートウェイ	あり	SSL証明書	あり	Gateway Server, HostServer, クライアント	すこし 難しい	管理者専用のサイトが用意されており、導入まで丁寧に記載されている
SoftEther VPN	レイヤ2でカプセル化やトンネリングを行う、VPN構築ソフトウェア	あり	認証 RSA 暗号化 AES、DES など	なし	Gateway Server, HostServer, クライアント	すこし 易しい	インストールまでのロードマップが用意されている コンピュータネットワークへの深い知識が必要

6.1 Teleport

概要

Teleport(<https://github.com/gravitational/teleport>) は、リモートアクセスのためのセキュリティ Gateway となっており、開発者によると、従来の OpenSSH の代わりに使用されることを目的としているという。

有料版と無料版があり、今回は無料版を使用した。以下に Telport の機能の一部を示す。

- 単一の SSH アクセス Gateway
- SSH 証明書ベースの認証

- 二段階認証
- SSH の役割ベースのアクセス制御
- セッションの記録を行う。
- シングルサインオン (SSO)

基本的なアーキテクチャの概要を下図に示す。

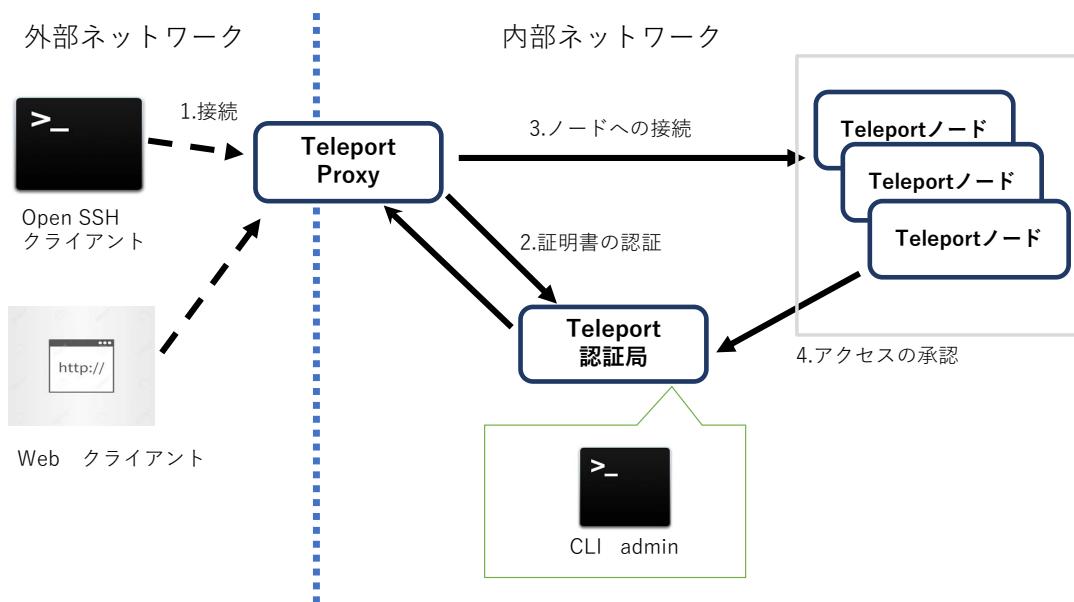


図 6.1: Teleport のアーキテクチャ

Teleport の構成要素

Teleport の仕組みを解説する前に、用語の説明を行う。

- ノード
「サーバ」または「コンピュータ」と同義語。SSH 接続できるもの。
- ユーザ
ノード上で一連の操作を実行できる人や、マシン。
- クラスター
連携して動作するノードのグループであり、単一のシステムとみなすことができる。
- 認証局 (CA)
公開鍵/秘密鍵のペアの形式で SSL 証明書を発行する。
- Teleport ノード
テレポートサービスを実行しているノード。許可されたテレポートユーザがアクセスできる。

- Teleport ユーザ
テレポートクラスタへアクセスしたいユーザ。ユーザはユーザ名とパスワードを登録する必要がある。
- Teleport クラスタ
1つ以上のノードで構成され、各ノードは同じ CA によって署名された証明書を持つ。
- Teleport 認証局
テレポートは、認証サービスの機能として2つの内部CAを運用する。一つはユーザ証明書の署名を行い、もう一つはノード証明書の署名に使用される。

Teleport サービス

Teleport では、「ノード」、「認証」、「プロキシ」の3つのサービスが連携して機能している。Teleport ノードは、SSH を使用してリモートでアクセスできるサーバーである。「OpenSSH」、「TeleportCLI クライアント (tsh)」、「Web ブラウザー」を使用することで Teleport ノードにログインすることができる。

Teleport 認証局は、ユーザとノードを認証し、ノードへのユーザーアクセスを承認し、ユーザーとノードに発行された証明書に署名することで認証局として機能する。

TeleportProxy は、ユーザー資格情報を認証サービスに転送し、認証が成功した後、要求されたノードへの接続を作成し、WebUI を提供する。

Teleport の仕組み

Teleport の隔離サーバへ接続までの流れ

下図の詳細なアーキテクチャを使って説明を行う。

1: クライアント接続を開始する

クライアントは、CLI インターフェースや Web ブラウザーを使用してプロキシ (Gateway) へ SSH 接続を始める。そのとき、クライアントは証明書を提供する。

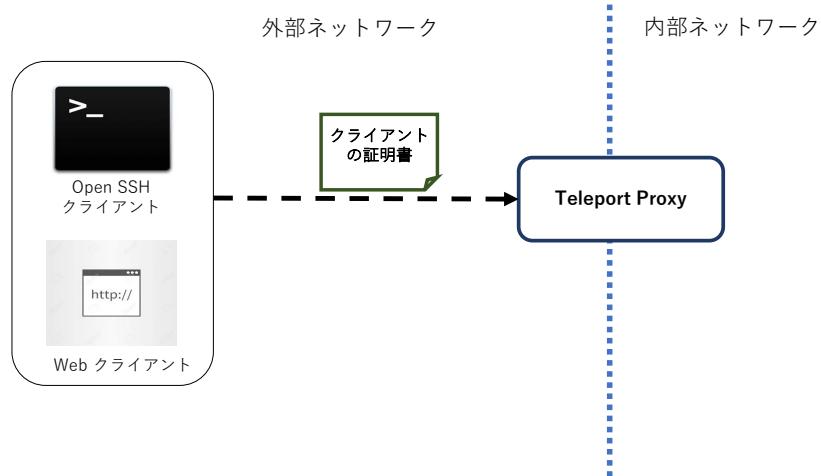


図 6.2: クライアント接続開始

2: クライアント証明書の認証

プロキシは、送信された証明書が以前に認証サーバ（CA）によって署名されているかどうかを確認する。もし署名がされていなかった場合（初回ログイン時）や証明書の有効期限が切れているとき、プロキシは接続を拒否し、パスワードと二段階認証でのログインをクライアントに求める。二段階認証は、Google Authenticatorなどを用いて行う。HTTPS 経由でプロキシに送信される。

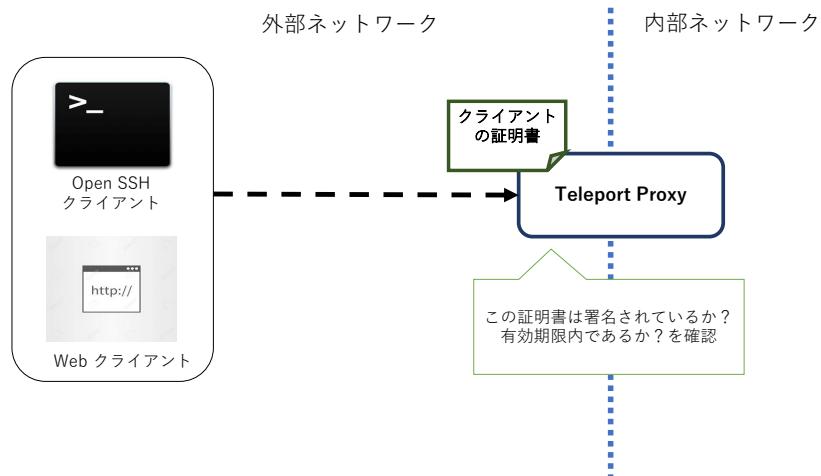


図 6.3: クライアント証明書の認証

3: クライアントが接続要求するテレポートノードを調べる。

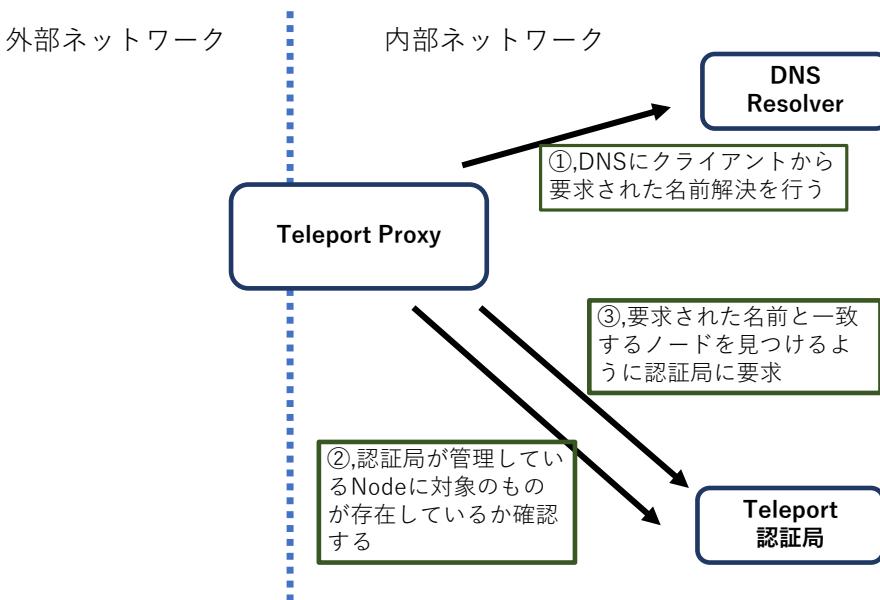


図 6.4: ノードの検索

このステップで、プロキシーはクラスター内の要求されたノードを見つけようとする。プロキシがノードのIPアドレスを見つける検索メカニズムは3パターンある。

- (1) DNSを使用して、クライアントから要求された名前解決を行う。
- (2) 登録されているノードがあるかどうか認証サーバに尋ねる。
- (3) 要求された名前と一致するラベルを持つノードを見つけるように認証サーバに要求する。

その後、ノードが見つかると、プロキシはクライアントと要求されたノード間の接続を確立する。その後、宛先ノードはセッションの記録を開始し、セッション履歴を認証サーバ(CA)に送信して保存する。

4: ノード証明書の認証

ノードは接続要求を受信すると、認証サーバを使用してノードの証明書を検証しノードのクラスタメンバーシップを検証する。ノード証明書が有効な場合、ノードは、クラスター内のノードおよびユーザーに関する情報へのアクセスを提供する認証サーバーAPIへのアクセスを許可される。

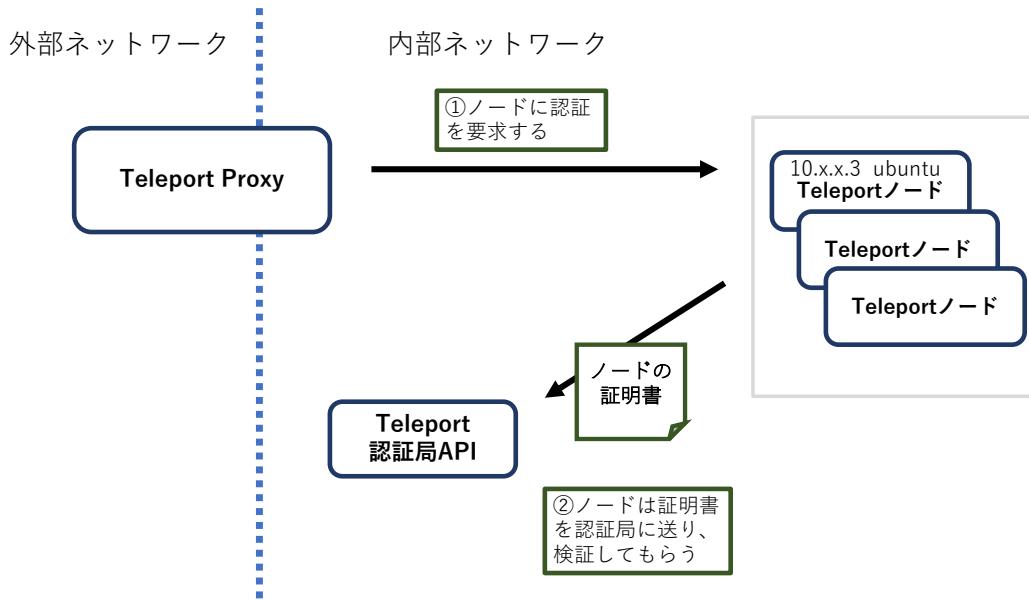


図 6.5: ノードの認証

5: ユーザノードのアクセスを許可する

ノードは認証サーバーに、接続しているクライアントの OS ユーザーのリスト（ユーザー マッピング）を提供するように要求し、クライアントが要求された OS ログインの使用を許可されていることを確認する。

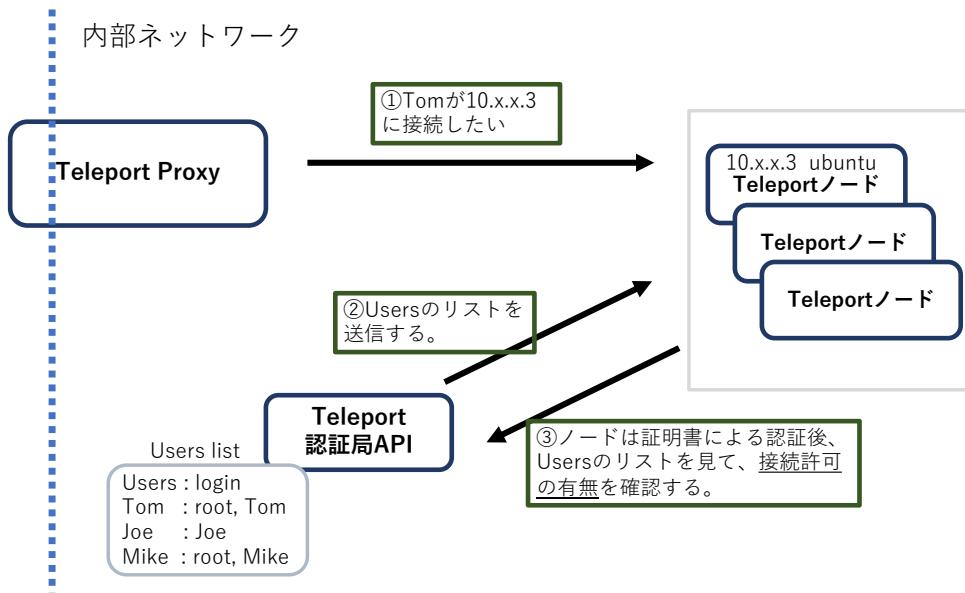


図 6.6: ノードへのアクセス許可

6: 最後に、クライアントはノードへの SSH 接続を作成することを許可され、目的のノードへ

ドへの SSH 接続を行う。

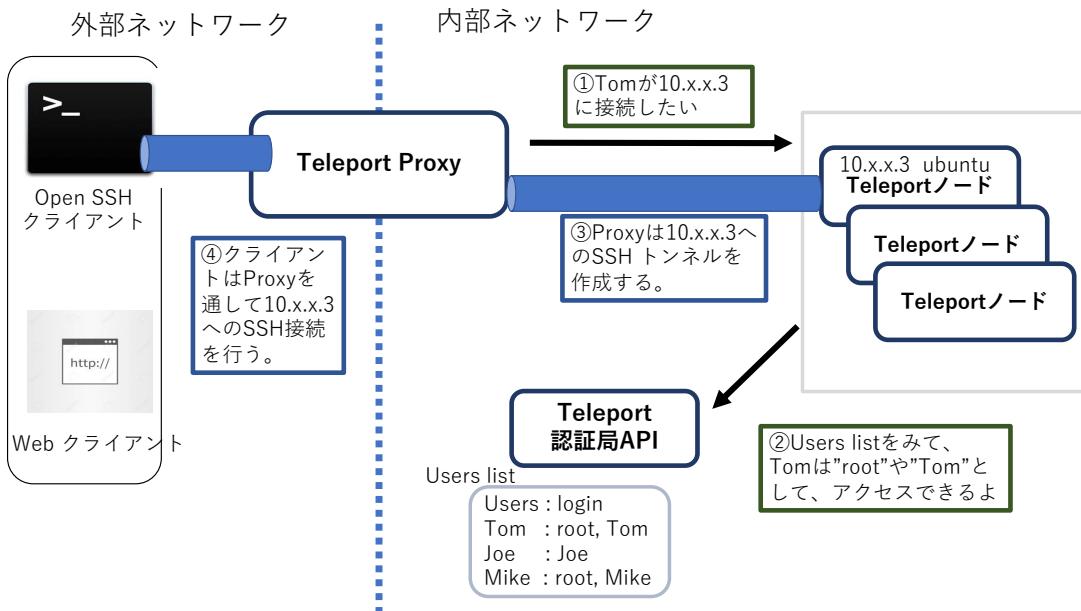


図 6.7: SSH 接続の完了

Teleport の評価

Teleport は、ProxyServer(Gateway)、AuthServer、遠隔アクセス先端末のそれぞれに対応したソフトウェアをインストールし、設定を行う必要がある。Teleport では、外部から Proxy へのアクセスがあった際、接続者の持つ SSH 証明書を AuthServer が検証し、Auth に登録された権限（管理者、アルバイト、開発者）に沿って、接続者が要求する隔離サーバへの Tunnel を引いている。Teleport を利用することで、誰に対してどのサーバにどんな権限でアクセスさせたいかを統合管理でき、シェル内で入力されたコマンドは全て記録されるため、誰がいつ何をしたのかを動画として確認することができる。

6.2 SoftEther VPN

SoftEther VPN とは

SoftEther VPN(<https://ja.softether.org/>) とは、筑波大学における学術目的の研究プロジェクト「SoftEther プロジェクト」により運営されている、オープンソースソフトウェアである。Windows,Linux,Mac,FreeBSD および Solaris 上で動作する。

SoftEther VPN の特徴

SoftEther VPN は、カプセル化およびトンネリングの通信をレイヤ 2、のデータリンク層で行なっている。SoftEther VPN を使用することで、通常の LAN カード、スイッチング HUB 及びレイヤ 3 スイッチなどのネットワークデバイスをソフトウェアによって仮想的に実現し、それらの間を TCP/IP プロトコルをベースとした、「SoftEther VPN プロトコル」と呼ばれるトンネルで接続することで、柔軟性の高い VPN 構築を実現している。

SoftEtherVPN の使い方

SoftEtherVPN は様々な種類の VPN の利用パターンを想定している。その一部を以下に挙げる。

- 企業内における VPN
 - PC 間接続 VPN
 - リモートアクセス VPN
 - 拠点間接続 VPN
- クラウドにおける VPN
 - ローカル PC をクラウドへ参加させる方法
 - クラウド VM を企業内 LAN に参加させる方法
 - クラウドと LAN のブリッジ VPN 接続
 - 複数クラウド間の VPN ブリッジ接続
- モバイルにおける VPN
 - iPhone および Android
 - Windows や Mac のモバイル PC

本研究では、「リモートアクセス VPN」用に SoftEtherVPN Server、Client を各種インストールし、設定を行う。

6.2.1 リモートアクセス VPN (SoftEtherVPN)

SoftEtherVPN には、「VPN Server」、「VPN Client」、「VPN Bridge」の大きく 3 種類の要素がある。リモートアクセス VPN の実現には、「VPN Server」と「VPN Client」を用いる。

SoftEtherVPN を使用したリモートアクセス VPN での、イメージ図を下図 6.8 に記す。

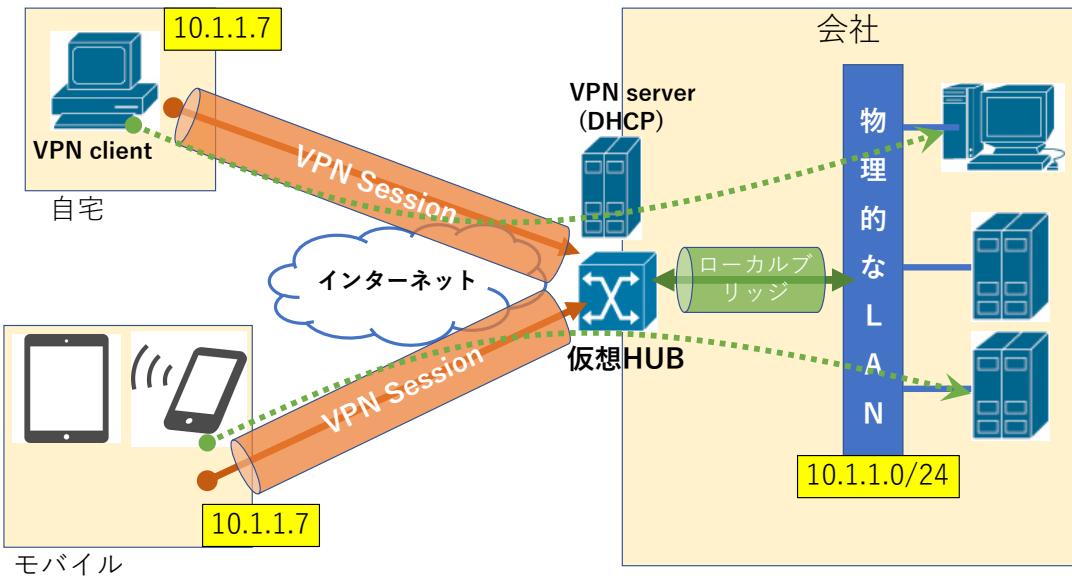


図 6.8: SoftEtherVPN のリモートアクセス利用

リモートアクセス VPN を構築するためには、仮想的なネットワークセグメントと物理的な Ethernet ネットワークセグメントとの間を「ローカルブリッジ接続機能」により接続する必要がある。そうすることにより、VPN を経由して仮想 HUB に接続下すべてのリモートコンピュータは、物理的な既存の Ethernet セグメントの一部として扱われるようになる。

ローカルブリッジとは

「ローカルブリッジ接続機能」を使用すると、VPN Server または VPN Bridge 内で動作している仮想 HUB と、そのサーバーコンピュータ上に接続されている物理的な LAN カードとの間をレイヤ 2 (データリンク層) で接続し、元々別々の Ethernet セグメントとして動作していた 2 つのセグメントを一つのセグメントに結合することができる。

ローカルブリッジにより、仮想 HUB に接続しているコンピュータと物理的な LAN に接続しているコンピュータの間で互いに相手が物理的には別のネットワークに接続されているにも関わらず、論理的には同一の Ethernet セグメントに接続されている事になり、Ethernet のレベルで自由に通信することができるようになる。

ローカルブリッジを使用することで、リモートアクセス型 VPN および拠点間接続型の VPN を簡単に構築することができる。

ローカルブリッジを使用する上での注意点として、ローカルブリッジ用に新しい物理的な LAN カードを増設する必要があるということである。ローカルブリッジ接続先として使用したい LAN カードが、その VPN Server や VPN Bridge として通常の通信に使用している LAN カードであってはいけないのである。

SoftEtherVPN のインストールとリモートアクセス VPN の構築

SoftEtherVPN には、「VPN Client」、「VPN Server」のそれぞれ別々の種類のソフトウェアが用意されている。そのため、それぞれに対するインストール方法と設定をここに記す。本研究でのネットワーク構成に合わせたリモートアクセス型 VPN の図を下図 6.9 に表す。

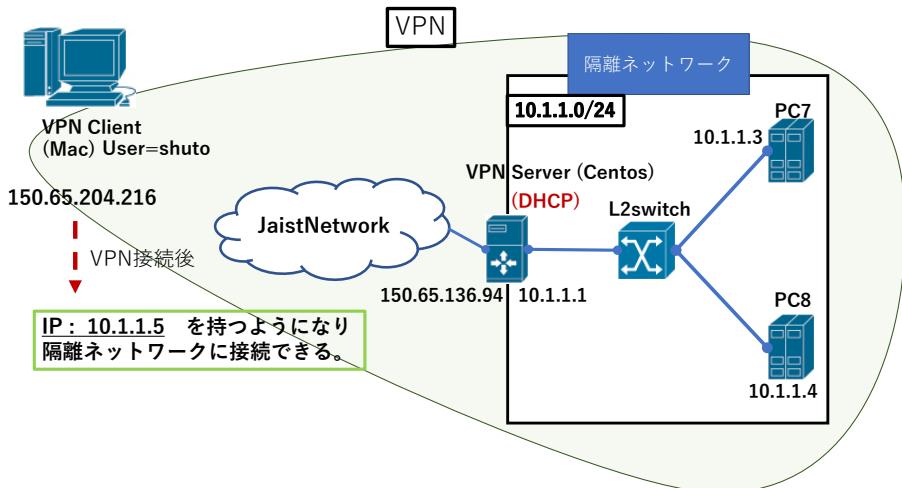


図 6.9: リモートアクセス VPN

SoftEther VPN Server の DHCP 設定

ここでは、リモートアクセス型の SoftEtherVPN の利用には DHCP サーバが必要であるため、上図 6.9 の VPNServer(Centos) を DHCP を行えるよう設定する。あらかじめ”SELinux”は無効化している。

まず、\$ sudo yum -y install dhcp で DHCP をインストールする。その後以下のように DHCP サーバ設定ファイル (/etc/dhcp/dhcpd.conf) を作成した。ここでは、10.1.1.0/24 で IP アドレスを自動で割り当ってくれるように記述した。

```
#  
# DHCP Server Configuration file.  
#   see /usr/share/doc/dhcp-server/dhcpd.conf.example  
#   see dhcpcd.conf(5) man page  
subnet 10.1.1.0 netmask 255.255.255.0 {  
    range 10.1.1.3 10.1.1.254;  
    option routers 10.1.1.1;  
    option broadcast-address 10.0.0.255;  
    option domain-name-servers 10.1.1.1;  
}
```

- \$ sudo systemctl start dhcpcd
(DHCP サーバの起動)

- \$ sudo systemctl enable dhcpcd
(DHCP サーバの自動起動設定)

その後、クライアント側の DHCP を ON にすることで、IP アドレスの自動割当が行える。

SoftEther VPN Server のインストールと設定

ここでは、まず VPNServer 側のインストールと設定をまとめます。

まず、<https://www.softether-download.com/ja.aspx?product=softether> から linux 用のものをダウンロードする。

1. \$ tar -xzvf softether-vpnclient-v4.28-9669-beta-2018.09.11-linux-x64-64bit.tar.gz
(解凍)
2. \$ cd vpnclient/
3. \$ make
(make コマンドでビルドを行う。ライセンスの同意を求められるため、1 を入力して同意する。)
4. \$ cd .. /
5. \$ sudo mv vpnclient/ /usr/local/
(生成されたディレクトリを /usr/local/ に移動する。)
6. \$ cd /usr/local
7. \$ sudo chown -R root:root vpnclient
8. \$ cd vpnclient
9. \$ sudo chmod 600 *
10. \$ sudo chmod 700 vpncmd
11. \$ sudo chmod 700 vpnclient
(パーミッションを設定する。)

サーバー側に、サービスファイル (/etc/systemd/system/vpnserver.service) を定義する。

```
[Unit]
Description=SoftEther VPN Server
After=network.target network-online.target

[Service]
ExecStart=/usr/local/vpnserver/vpnserver start
ExecStop=/usr/local/vpnserver/vpnserver stop
Type=forking
RestartSec=3s

[Install]
WantedBy=multi-user.target
```

systemctl コマンドでサービスの開始と有効化を行う。

1. \$ sudo systemctl start vpnserver

```
2. $ sudo systemctl enable vpnserver
```

次に SoftEther VPN Server の設定を行う。設定はコマンドラインツール「vpncmd」を使って設定した。ネットワークが繋がっていれば、vpncmd を使ってリモート設定も行える。

```
vpncmd コマンド - SoftEther VPN コマンドライン管理ユーティリティ
SoftEther VPN コマンドライン管理ユーティリティ (vpncmd コマンド)
Version 4.34 Build 9745 (Japanese)
Compiled 2020/04/05 23:39:56 by buildsan at crosswin
Copyright (c) SoftEther VPN Project. All Rights Reserved.

vpncmd プログラムを使って以下のことができます。

1. VPN Server または VPN Bridge の管理
2. VPN Client の管理
3. VPN Tools コマンドの使用 (証明書作成や通信速度測定)

1 - 3 を選択: 1

接続先の VPN Server または VPN Bridge が動作しているコンピュータの IP アドレスまたはホスト名を指定 'ホスト名:ポート番号' の形式で指定すると、ポート番号も指定できます。
(ポート番号を指定しない場合は 443 が使用されます。)
何も入力せずに Enter を押すと、localhost (このコンピュータ) のポート 443 に接続します。
接続先のホスト名または IP アドレス: 150.65.136.94:992

サーバーに仮想 HUB 管理モードで接続する場合は、仮想 HUB 名を入力してください。
サーバー管理モードで接続する場合は、何も入力せずに Enter を押してください。
接続先の仮想 HUB 名を入力:
パスワード: *****

VPN Server "150.65.136.94" (ポート 992) に接続しました。

VPN Server 全体の管理権限があります。

VPN Server>■
```

図 6.10: vpncmd 起動画面 (Server)

vpncmd 起動後は、Help を参照しながら必要に応じて拡張機能の各種設定を行う。

```
VPN Server> ServerPasswordSet
          (管理者パスワード設定)
VPN Server> BridgeCreate
          (ローカルブリッジ接続を作成する)
VPN Server> IPsecEnable
          (L2TP/IPSec サーバーを有効化する)
```

次に、仮想 HUB の設定を行う。本研究では、仮想 HUB 名を「Subthemehub」として使用するユーザーを「shuto」とした。vpncmd で仮想 HUB 設定モードに移行するには、起動時に仮想 HUB 名を入力するか、Server 設定モードで以下のコマンド入力すれば良い。

```
VPN Server> hublist
          (仮想 HUB 一覧出力)
VPN Server> hubCreate subthemehub
          (subthemehub という名前の仮想 HUB を作成する)
```

VPN Server> hub subthemehub
(subthemehub の設定モードに移行する)
VPN Server/subthemehub>

仮想 HUB の新規作成ができたら、仮想 HUB 設定モードで Softether VPN を利用するユーザの登録を行う。

VPN Server/subthemehub> userlist(ユーザーの一覧出力)

VPN Server> userCreate shuto
(shuto という名前のユーザーを作成する)
VPN Server> userpasswordset shuto
(shuto の認証をパスワード認証に切り替える)

SoftEther VPN Client のインストールと設定

SoftEther VPN Client のインストール方法は、上記の Server とほぼ同じコマンドであり、server を client に置き換えるだけである。MacOS で VPN Client を使用するには、tuntap のインストールが必須である。homebrew で簡単にインストールできる。tuntap は仮想 LAN カードの作成に必要になる。

SoftEther VPN client の起動は、centos の場合は VPN server と同じコマンドだが、今回使用した Mac では、\$ sudo ./vpncmd start と入力する必要がある。

次に、VPN Client の設定を記す。VPN Server と同じように「vpncmd」を使ってコマンドラインで設定を行なった。

```
sh-3.2# ./vpncmd
vpncmd コマンド - SoftEther VPN コマンドライン管理ユーティリティ
SoftEther VPN コマンドライン管理ユーティリティ (vpncmd コマンド)
Version 4.34 Build 9745 (Japanese)
Compiled 2020/04/05 23:39:56 by buildsan at crosswin
Copyright (c) SoftEther VPN Project. All Rights Reserved.

vpncmd プログラムを使って以下のことことができます。

1. VPN Server または VPN Bridge の管理
2. VPN Client の管理
3. VPN Tools コマンドの使用 (証明書作成や通信速度測定)

1 - 3 を選択: 2

接続先の VPN Client が動作しているコンピュータの IP アドレスまたはホスト名を指定してください。
何も入力せずに Enter を押すと、localhost (このコンピュータ) に接続します。
接続先のホスト名または IP アドレス:

アクセスが拒否されました。パスワードが間違っているか、接続する管理モードが正しくない可能性があ
う一度入力することができます。キャンセルする場合は、Ctrl + D を押してください。
パスワード: *****

VPN Client "localhost" に接続しました。

VPN Client>
```

図 6.11: vpncmd 起動画面 (Client)

VPNClient では、始めに利用する仮想 LAN カードを作成する必要がある。本研究では「SubthemeNic」という名前の LAN カードを作成した。

VPN Client> NicCreate SubthemeNic (新規仮想 LAN カードの作成)

仮想 LAN カードの作成が完了したら、次に接続設定を行う。

```
VPN Client>accountcreate
AccountCreate コマンド - 新しい接続設定の作成
接続設定の名前: subtheme

接続先 VPN Server のホスト名とポート番号: 150.65.136.94:992

接続先仮想 HUB 名: subthemehub

接続するユーザー名: shuto

使用する仮想 LAN カード名: subthemenic

コマンドは正常に終了しました。

VPN Client>
```

図 6.12: 接続設定作成画面

VPN Server での設定でパスワード認証を設定したため、VPN Client でも同じパスワードを登録する必要がある。

VPN Client> AccountPasswordSet Subtheme
これにより VPNClient の設定が完了した。

SoftEtherVPN Server への接続

最後に、VPNClient から、VPNServer に接続する。

VPN Client> accountconnect subtheme (使用する接続設定を指定)

```
$ sudo ipconfig set tap0 dhcp
(ここから vpncmd を抜けて、コマンドラインで行う。この tap0 とは接続設定で使用
した NIC と同じ MAC アドレスを持つものを指定する)
$ ifconfig tap0
```

```

sh-3.2# ifconfig tap0
tap0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 5e:67:3d:4d:6b:d9
      inet 10.1.1.7 netmask 0xffffffff broadcast 10.1.1.255
        media: autoselect
        status: active
        open (pid 14551)

```

図 6.13: ifconfig tap0 の出力
結果、隔離ネットワークと同じセグメントの IP アドレスを取得できた。

そして、最後に VPN Client>AccountStatusGet subtheme で接続状況を確認すると以下 6.14 の出力が得られた。これを見ると、暗号スイートが”TLS_AES_256_GCM_SHA384”をしており、プロトコル詳細を見ると TLS には OpenSSL を使用して、通信内容の暗号化に chachah20 という共通鍵暗号と Poly1305 というメッセージ認証方式を使用していることがわかる。

接続設定名	subtheme
セッション接続状態	接続完了 (セッション確立済み)
VLAN ID	-
サーバー名	150.65.136.94
ポート番号	TCP ポート 992
サーバー製品名	SoftEther VPN Server (64 bit)
サーバーバージョン	4.34
サーバービルド番号	Build 9745
接続開始時刻	2020年 9月30日(水) 16時29分 0秒
初回セッションの確立時刻	2020年 9月30日(水) 16時29分 0秒
現在のセッションの確立時刻	2020年 9月30日(水) 16時29分 0秒
セッション確立回数	1 回
半二重 TCP コネクションモード	いいえ (全二重モード)
VoIP / QoS 対応機能	有効 (使用中)
TCP コネクション数	2
TCP コネクション数最大値	2
暗号化の使用	はい (暗号化アルゴリズム: TLS_AES_256_GCM_SHA384)
圧縮の使用	いいえ (圧縮無し)
物理通信に使用中のプロトコル	Standard TCP/IP (IPv4)
プロトコル詳細	IPv4 UDPAccel_Ver=2 ChachaPoly_OpenSSL UDPAccel_MSS=1309
UDP 高速化機能をサポート	はい
UDP 高速化機能を使用中	はい
セッション名	SID-SHUTO-12
コネクション名	CID-1233
セッションキー (160bit)	7BD4BF00D4C030216FD0CC92E2294BE578E126D8
ブリッジ / ルータモード	いいえ
モニタリングモード	いいえ
送信データサイズ	23,321 バイト
受信データサイズ	22,915 バイト
送信ユニキャストパケット数	12 パケット
送信ユニキャスト合計サイズ	504 バイト
送信ブロードキャストパケット数	61 パケット
送信ブロードキャスト合計サイズ	8,734 バイト
受信ユニキャストパケット数	14 パケット
受信ユニキャスト合計サイズ	1,188 バイト
受信ブロードキャストパケット数	79 パケット
受信ブロードキャスト合計サイズ	8,752 バイト
コマンドは正常に終了しました。	

図 6.14: 接続状況の出力画面

SoftEther VPN の評価

SoftEtherVPN は、GatewayServer、利用する Client、のそれぞれに対応したソフトウェア（VPNServer、VPNClient、VPNBridge）をインストールし設定を行う必要がある。しかし、Github にだけでなく専用のサイトが存在しそこに OS ごとの詳細な説明が記載されているとともに、日本発祥のソフトウェアであることにより”Qiita”などにも多くの情報が存在するため、とても容易にインストールと設定を行うことができた。今回は、認証方式をパスワード認証にしたが、その他にも RADIUS 認証やサーバ証明書認証なども行える。また、今回利用した”リモートアクセス型”だけでなく、その他十種類以上の利用形態が想定されており、とても安全性、柔軟性、拡張性に富んだソフトウェアといえる。

デメリットとしては、SoftEtherVPNServer をインストールするサーバが DHCP サーバであることが必須であることだ。これにより個人的に利用する場合は GatewayServer を各自 DHCP サーバ化する必要がある。サーバ等の知識が必要がなってくる。

第7章 通信制御に関する考察

本論文では、前章までにかけて通信制御や遠隔アクセスを容易にするアプリケーションを試用してきた。この章では、通信接続やその接続の管理を容易にするために必要な機能とそれを実現するための技術を考える。自分が考える必要な機能と実現する技術を以下に挙げ、それを以下表 7.1 と表 7.2 にまとめた。

1. 踏み台数（段数）を減らすこと。
2. 接続の際のトンネルの管理の容易にしたい。
3. ログイン時に入力の手間を省きたい。
4. ユーザ情報の一括管理が行える。
5. 初期登録時、一括管理に加えて登録の自動化を行える。
6. GUI が利用できる。
7. 接続ログやセッションログが取れたら良い
8. 利用端末と遠隔アクセス端末側 (E2E) のみアカウントがあれば利用できる。

表 7.1: 求める機能と技術とソフトウェア

求める機能	実現する技術	ソフトウェア
①踏み台数(段数)の削減	VPN トンネリング	sshuttle SoftEtherVPN
②トンネルの管理の容易化	特になし	該当なし
③ログイン時の入力作業の削減	シングルサインオン (Kerberos) 公開鍵、証明書認証 (SSH)	試用した全てのソフトウェア
④ユーザ情報の一括管理	LDAP Kerberos RADIUS	Active Directory sshportal Teleport SoftEtherVPN
⑤情報登録の自動化	特になし	該当なし
⑥GUI	特になし	Teleport SoftEtherVPN
⑦ログの取得	特になし	sshportal Teleport SoftEtherVPN
⑧E2Eのみのログイン情報による利用	VPN	sshuttle SoftEtherVPN Teleport

表 7.2: 求める機能とソフトウェアの対応表

	sshuttle	sshportal	Teleport	SoftEtherVPN	Active Directory
①踏み台数(段数)の削減	✓ VPN	—	—	✓ VPN	—
②トンネルの管理の容易化	—	—	—	—	—
③ログイン時の入力作業の削減	✓	✓	✓	✓	✓
④ユーザ情報の一括管理	—	✓ LDAP	✓ RADIUS LDAP	✓ RADIUS LDAP	✓ LDAP Kerberos
⑤情報登録の自動化	—	—	—	—	—
⑥GUI	—	—	✓	✓	—
⑦ログの取得	—	✓	✓	✓	—
⑧E2Eのみのログイン情報による利用	✓ VPN	—	—	✓ VPN	—

踏み台数（段数）を減らす

システム管理者やユーザにとって、外部から隔離ネットワーク内のサーバにアクセスする際に最も煩わしく感じる場面は、例えばいくつもの Gateway を経由して、多段 SSH しながら目的のサーバに接続を行う作業であろう。ホスト名とユーザ名を指定して何度もアカウント情報とコマンドを入力するのは大変面倒なことである。このような場面で必要となる最も代表的な技術は 2.5 節で説明した「VPN」である。VPN を設定することで、図 7.1 のように利用する端末と接続先の端末の存在するネットワーク間で仮想的なネットワークを引くことができ、殆どの場合一度の接続コマンドで目的を達成できる。

本論文で試用したソフトウェアで同様のことができるものは、以下の 2 つである。

- sshuttle (5.1 節)
- SoftEtherVPN (6.2 節)

「sshuttle」は、遠隔アクセスを行う端末にインストールするだけで、sshuttle コマンドと引数に Gateway のホスト名か IP アドレスと隔離ネットワークのサブネットを指定することで隔離ネットワークに対して簡易的な VPN を形成してくれるソフトウェアである。

「SoftEtherVPN」は、遠隔アクセス端末と接続先端末の両方に対応したコンポーネントを適用することで、レイヤ 2 での VPN を実現するソフトウェアである。SoftEther を利用することで実際に隔離ネットワーク内の IP アドレスを割り当ててもらうことができ、実際の Ethernet に接続されているように実感することができる。

「VPN」によって、接続先サーバと仮想的に同じネットワークに所属させることで接続コマンド入力の回数を減らす方法の他に、2.2 節で説明したように「トンネリング」よって回数を減らすこともできる。隔離サーバと Gateway 間、利用端末と Gateway 間にトンネルを引くことで少ない段数で接続を完了することができる。

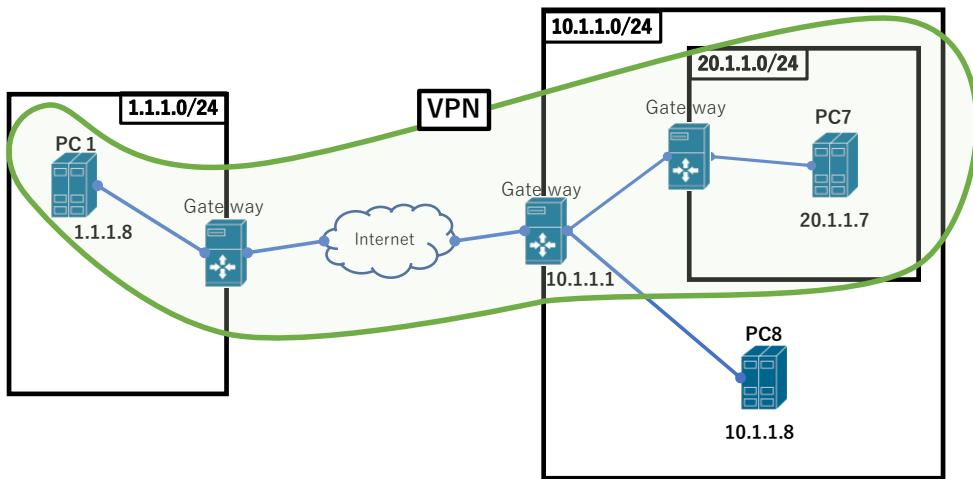


図 7.1: VPN イメージ

本論文で試用したソフトウェアで同様のことができるものは、以下の 4 つである。

- sshuttle (5.1 節)
- sshportal (5.2 節)
- Teleport (6.1 節)

「Teleport」では、認証を行った後隔離サーバと Gateway 間でトンネルが引かれている。これらの他に、OpenSSH もトンネリングを行うことができる。

トンネルの管理の容易化

前述したように、接続の段数を減らすために「トンネリング」の技術が用いられる。段数を減らすことができるが、最初のアクセス時には目的サーバまでのトンネルを引く必要がある。そのため、ユーザは対応するコマンドを入力することでトンネルを作成し、目的地までそのコマンド何度も入力するのは面倒である。また、システム管理者は Gateway を経由して引かれたトンネルを管理する必要がある。使われなくなったトンネルを切断したり、悪意のある使用がされているのか、誰が使用しているかなどを知る必要がある。

今回試用したソフトウェアで「トンネルの管理を容易化」の機能に該当するものはなかった。

ログイン時に入力の手間

ユーザや管理者にとって、遠隔アクセスやサーバ利用の際に上記の「接続先への段数を減らす」ことの他に、「ログイン時の入力の手間」を削減できるのであれば、とても便利

なことであろう。「接続先への段数を減らす」ことができたとしても、他のサーバアクセスや機器利用の際に再びパスワード等の情報を入力するのは大変面倒である。

このような場面で必要となる技術はいくつかあるが、「Kerberos 認証による、シングルサインオン(SSO)」や「SSH 公開鍵認証の利用」、「SSH 証明書認証の利用」である。3.3 節で解説したように、Kerberos 認証を用いることでシングルサインオン(SSO)を実現でき、一度のログインのみで範囲内のサーバや機器に自由にアクセスすることができる。他のテクニックとして、SSH において公開鍵暗号を遠隔アクセス利用端末で作成し、接続先端末に登録しておけばアカウントを入力する手間を省くことができる。同様に、SSH で証明書認証を利用することで、SSH コマンドを入力すれば自動的にログインすることができる。

今回試用した全てのソフトウェアで、同様のことができる。

ユーザの情報の一括管理

システム管理者にとって面倒な作業だと感じる瞬間の 1 つは、一括管理されていないサーバに新規ユーザ登録の時であろう。一括管理されていないために、1 台 1 台に root 権限で入ったのち新規ユーザの登録を行わなければならない。そして、ユーザからすれば常に同じサーバのところに赴きログインしなければいけなくなる。なぜなら、異なるサーバだとアカウント情報が登録されてないからである。

今サーバ利用の例を出したが、LAN への接続利用の場合も似たようなことがいえる。LAN 接続のためのアカウント情報が一括管理されていない場合は、ユーザは学内や社内 LAN に接続する際に、アカウントが登録されている機器にしか接続できなくなってしまう。

これらの場面で必要になる技術は、3 章の中で解説した、「LDAP」や「Kerberos」、「RADIUS」である。「LDAP」、「Kerberos」を利用してことで、サーバのアカウント情報を一括管理することができる。これにより、管理者は LDAP サーバ、KDC サーバにユーザ情報を登録するだけで、範囲内のサーバにアカウント情報を適用させることができ、ユーザはどのサーバからでもログインすることができる。また、ネットワーク接続においては、「RADIUS」を利用してことでネットワークアカウント情報を一括管理でき、ユーザはどの機器からもネットワークに接続することができる。加えて、「LDAP」と「RADIUS」を連携させることで、サーバのアカウント情報をネットワークアカウント情報と兼ねることができる。

本論文で試用したソフトウェアで同様のことができるものは、以下の 4 つである。

- Active Directory (3.4 節)
- sshportal (5.2 節)
- Teleport (6.1 節)
- SoftEtherVPN (6.2 節)

「sshportal」では、「LDAP」等の技術を利用してデータベースにアカウント情報を登録することでユーザのログイン認証を行っている。開発者によると、今後 sshportal v2 において LDAP などの認証技術を利用できるようにするようである。

「Active Directory」では、ディレクトリーサービスの実現に「LDAP」、ユーザ認証の実現に「Kerberos」の技術を用いている。「Teleport」、「SoftEtherVPN」では、オプションとして「Active Directory」や「RADIUS」と連携できるようになっている。

一括管理における登録の自動化

前述のアカウント情報の一括管理に加えて、情報の登録作業を自動化できるのならば、管理者にとってとても便利なことであろう。管理者が何十人、何百もの情報を登録しなければならない状況であった時、1人1人手打ち入力で登録していくのはとても面倒な作業である。

そういう場合の「技術」というものは見つからなかった。調べた中にあったものは、シェルスクリプトを用いて一度に行うものや、エクセルを用いてできるだけ入力作業を減らすテクニックは存在した。

GUI が利用できる

一般的に管理者は CUI で作業をすることが多い。大規模システムにおいて、CUI での作業はシステムの一括制御がしやすくなるためである。しかし、CUI だけでなく GUI も利用することができるのならば、視覚的にログイン状況や、サーバの利用率、負荷を確認できるようになり、より管理しやすくなるだろう。

今回試用したソフトウェアで、GUI を実現している 2 つを以下にあげる。

- Teleport (6.1 節)
- SoftEtherVPN (6.2 節)

「Teleport」では、クライアントから接続可能なサーバのリストを表示したり、それらに対しての Web ベースのターミナルを開いたり、記録されたセッションを表示したり再生したりすることができる。

接続ログやセッションログの取得

管理者にとって、外部から隔離ネットワークに接続してくるユーザのアクセスログを取得できることは、セキュリティの向上につなげることができる。

本論文で試用したソフトウェアで同様のことができるものは、以下の 3 つである。

- sshportal (5.2 節)
- Teleport (6.1 節)
- SoftEtherVPN (6.2 節)

エンドツーエンド（E2E）のみのアカウント情報の利用

外部から隔離ネットワーク内のサーバにアクセスする際に、経由する全てのサーバのアカウント情報知る必要があるのは大変煩わしいことである。遠隔アクセス利用端末と接続先端末の、エンドツーエンドに関する情報だけ知ればよくなれば大変便利である。このような場面で必要となる技術は、「SSH の段数を減らす」際に用いられた、「VPN」技術である。VPN を用いることで、利用端末と接続先端末間で仮想ネットワークを構築するため、遠隔アクセスには 2 つのアカウント情報のみで十分となる。

本論文で試用したソフトウェアで同様のことができるものは、以下の 2 つである。

- sshuttle (5.1 節)
- SoftEtherVPN (6.2 節)

第8章　まとめ

本論文では前章にかけて、遠隔アクセスのために使われる技術とそのセキュリティ性向上のために使われている技術を解説した後、それらの技術を用いていくつかのオープンソースソフトウェアを試用し、通信制御に関する考察を行った。本章では全体のまとめを行う。

本研究では、ソフトウェアを試用するために3台のサーバと1台のL2スイッチで隔離ネットワークを構成しなければならなかった。サーバにはそれぞれにOSを入れ、スイッチには適した設定自分で行い、ソフトウェアを1つ1つインストールしていき、性能を調査した。インストールするソフトウェアは「Github」から、できるだけお気に入りやフォークが多いものを選んだ。研究開始時、お気に入り数が少ないものも選んでいたが、お気に入り数が少ないものは個人目的のために作られているもの、ソフトウェアの更新がだいぶ昔に止まっているもの、エラーの多いものや、インストール方法や設定が記されていないものがほとんどであったため、結局それらお気に入り数の少ないものを除外し多くの人の目に止まっているソフトウェアを選択した。

前章の考察において羅列した、求める機能を最も多く満たしていたソフトウェアは「Teleport」と「SoftEtherVPN」であった。これら2つは大規模ソフトウェアという、Gatewayやクライアントそれに適した設定を行う必要があり、導入難易度が比較的高いものであつたため、それ相応の高い性能をもっていたと思われる。

いくつかソフトウェアを試用する中で個人的にとても便利だと感じたのは、「sshuttle」と「SoftEtherVPN」である。「sshuttle」は、簡易的なVPNを生成できるツールであるが、利用するには遠隔アクセスを行う端末側のみにインストールすればよく、Gatewayにsshuttle用の設定を行う必要がないため、すぐに「sshuttle」の利便性を感じることができた。

「SoftEtherVPN」は、大規模ソフトウェアの中では比較的容易に設定を行うことができる上に、高い性能を持っているソフトウェアであった。レイヤ2で機能するものであるため、他のVPNソフトウェアとは違う使用感（実際にIPを割り振られるところなど）を感じることができた。今回の使用方法の他にもさまざまな使用モデルが想定されており、海外から日本国内の自宅のネットワークを利用できるようにするなど、大変興味深い使用方法が記載されており、将来的に使いこなせるようになりたいと感じた。

参考文献

- [1] 日経新聞, 2020/7/26. 「西村経財相「在宅勤務を7割に」 経済界に再要請へ」.
- [2] 総務省. 総務省テレワークセキュリティガイドライン, 平成30年4月.
- [3] S. Kent and R. Atkinson. IP Authentication Header, RFC2402. November 1998.
- [4] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP), RFC2406. November 1998.
- [5] D. Harkins and D. Carrel. The Internet Key Exchange (IKE), RFC2409. November 1998.
- [6] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-Point Tunneling Protocol (PPTP), RFC2637. July 1999.
- [7] W. Simpson and Ed. The Point-to-Point Protocol (PPP), RFC1661. July 1994.
- [8] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer Two Tunneling Protocol "L2TP", RFC2661. August 1999.
- [9] A. Valencia, M. Littlewood, and T. Kolar. Cisco Layer Two Forwarding (Protocol) "L2F", RFC2341. May 1998.
- [10] J.T. Melvin and R.W. Watson. First Cut at a Proposed Telnet Protocol, RFC97. February 1971.
- [11] T. Ylonen, C. Lonvick, and Ed. The Secure Shell (SSH) Transport Layer Protocol, RFC4253. January 2006.
- [12] J. Sermersheim and Ed. Lightweight Directory Access Protocol (LDAP): The Protocol, RFC4511. June 2006.
- [13] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5), RFC4120. July 2005.
- [14] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS), RFC2865. June 2000.