Security, OS and VM.
YAMA Laboratory

SSLWatcher: SSL/TLS通信を監視し警告するハイパバイザ

平井 成海†

†電気通信大学

目次

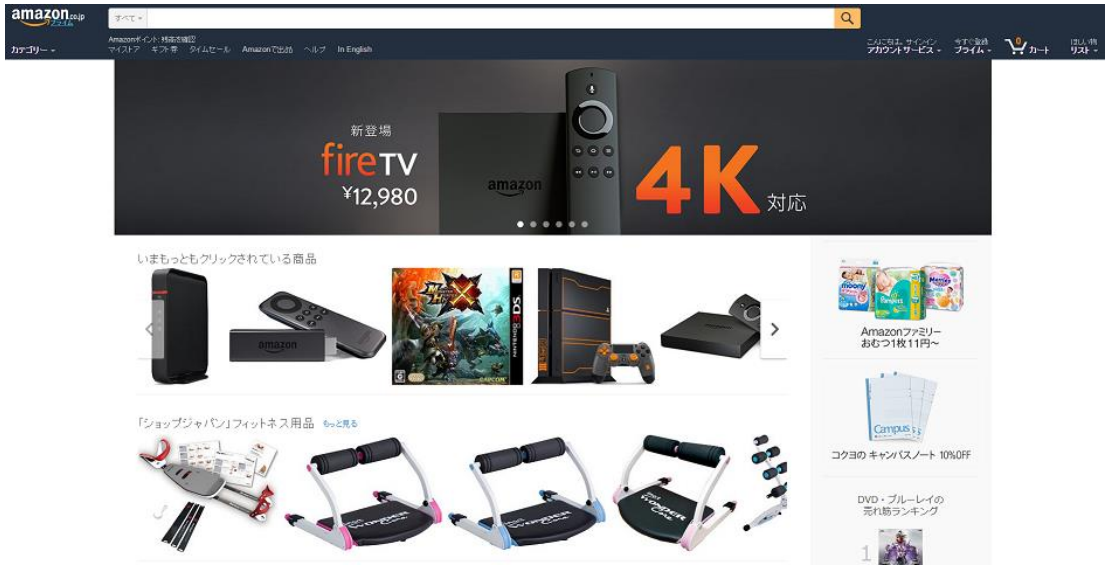
1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

目次

1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

背景

- 電子商取引やオンラインバンキングでは一般的にSSL/TLSが使用される
  - ▶ 機密性や完全性の確保
  - ▶ サーバ証明書による真正性の証明



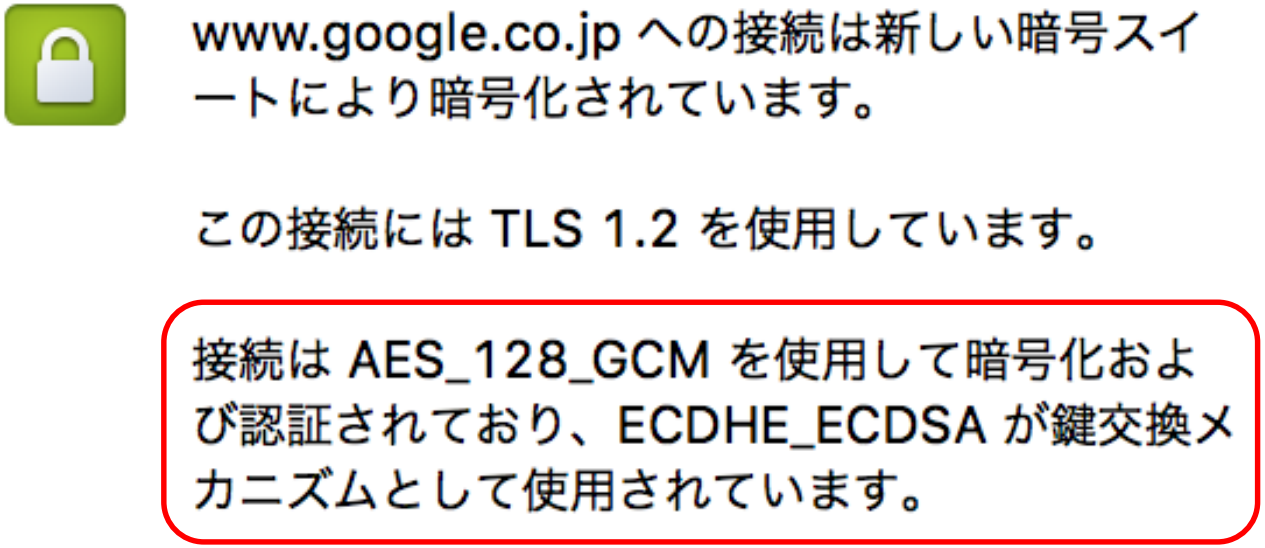
電子商取引サイトの例

背景

- しかし近年、SSL/TLSのセキュリティに関する問題が出てきている
  - 脆弱性や不適切なサーバ設定が原因
  - 脆弱性の例
    - FREAK (CVE-2015-0204)
    - Logjam (CVE-2015-4000)
  - 不適切なサーバ設定の例
    - 受け入れる暗号スイート(暗号アルゴリズムとハッシュアルゴリズムの組み合わせ)の不適切な設定

背景

- しかし近年、SSL/TLSのセキュリティに関する問題が出てきている
  - 画像はアプリケーション上の暗号スイートの表示だが、これを見てすぐにユーザは安全であると判断できない



背景

- しかし近年、SSL/TLSのセキュリティに関する問題が出てきている
  - 弱い暗号スイートをユーザが個別に拒否するのは難しい
    - 通常SSL/TLSのバージョン別でしか拒否できない(Internet Explorerなど)
    - ブラウザ以外のSSL/TLSを使用するアプリではSSL/TLSに関する設定が無い場合もある

目的と方針

- 安全性が損なわれるようなSSL/TLS通信を検出し警告することでユーザを守るようなシステムを作成する
  - ▶ 弱い暗号スイートや弱い鍵交換を検出する
- BitVisorを使用して通信を監視し警告対象の通信を見つけたら警告を行う
- 提案システムをSSLWatcherと呼ぶ

ユースケース

- SSL/TLS通信のセキュリティを向上させたいと考えている企業の管理者と従業員が対象

従業員用PC

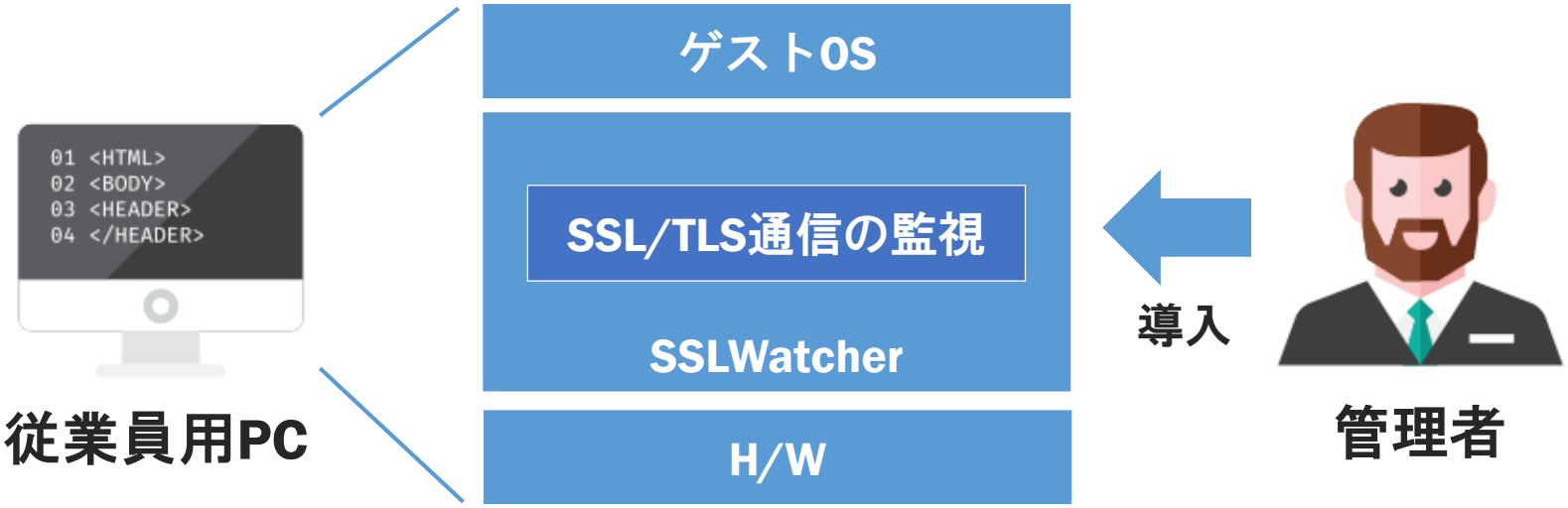
OS

H/W

管理者

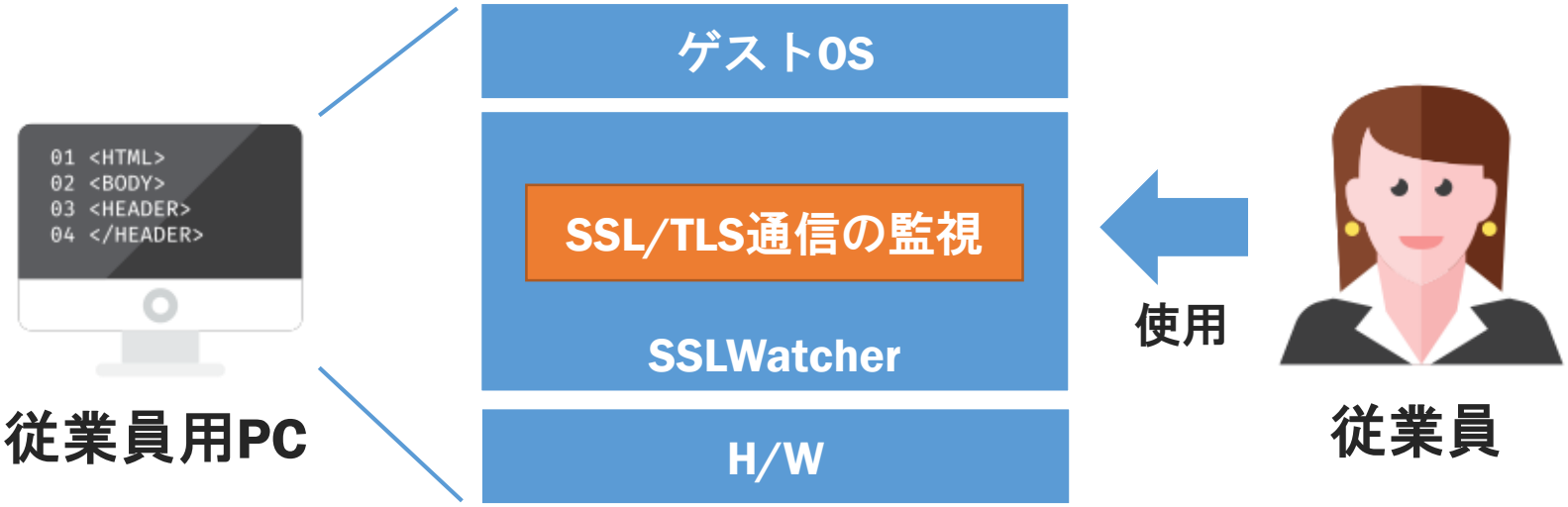
ユースケース

- SSL/TLS通信のセキュリティを向上させたいと考えている企業の管理者と従業員が対象



ユースケース

- SSL/TLS通信のセキュリティを向上させたいと考えている企業の管理者と従業員が対象



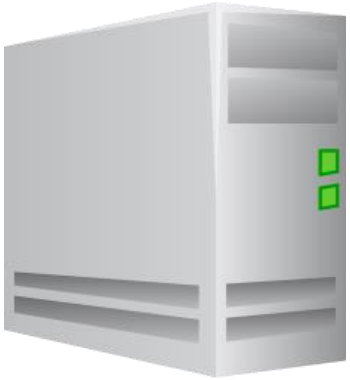
目次

1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

SSL/TLSのハンドシェイク

ClientHello

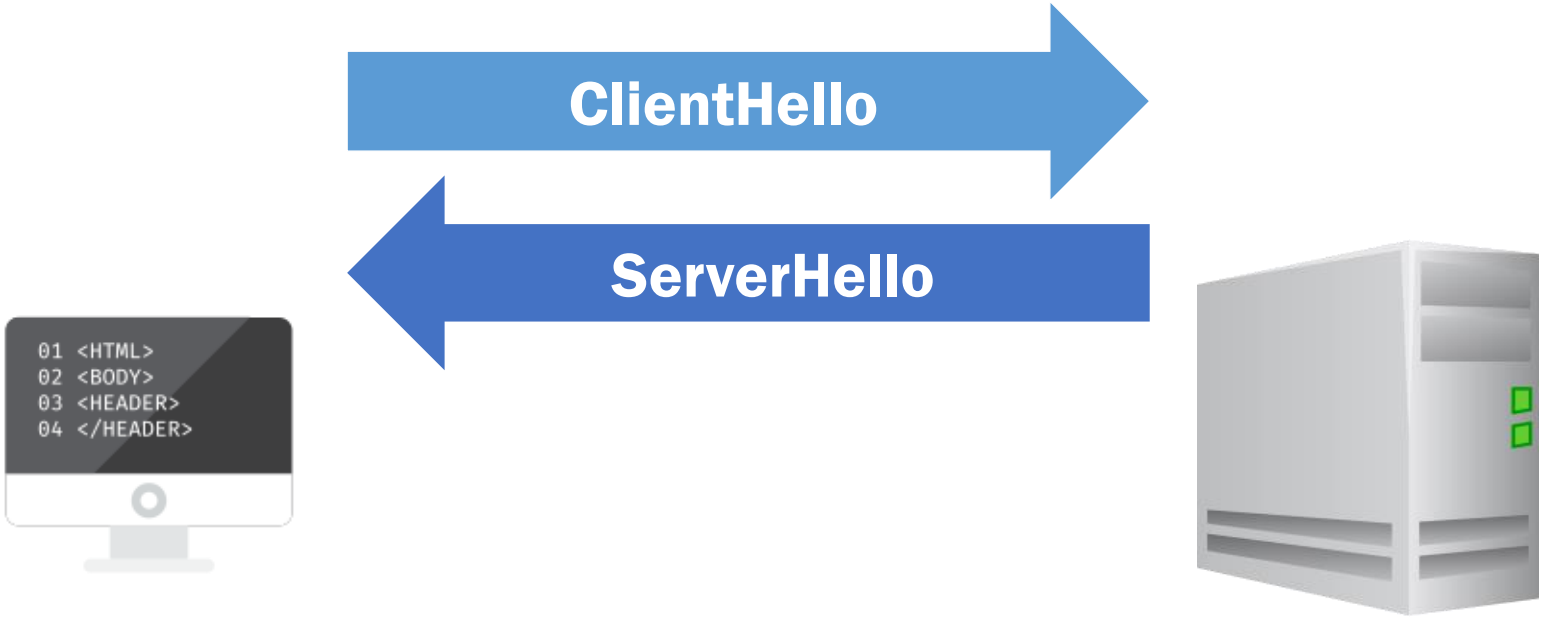
クライアント



サーバ

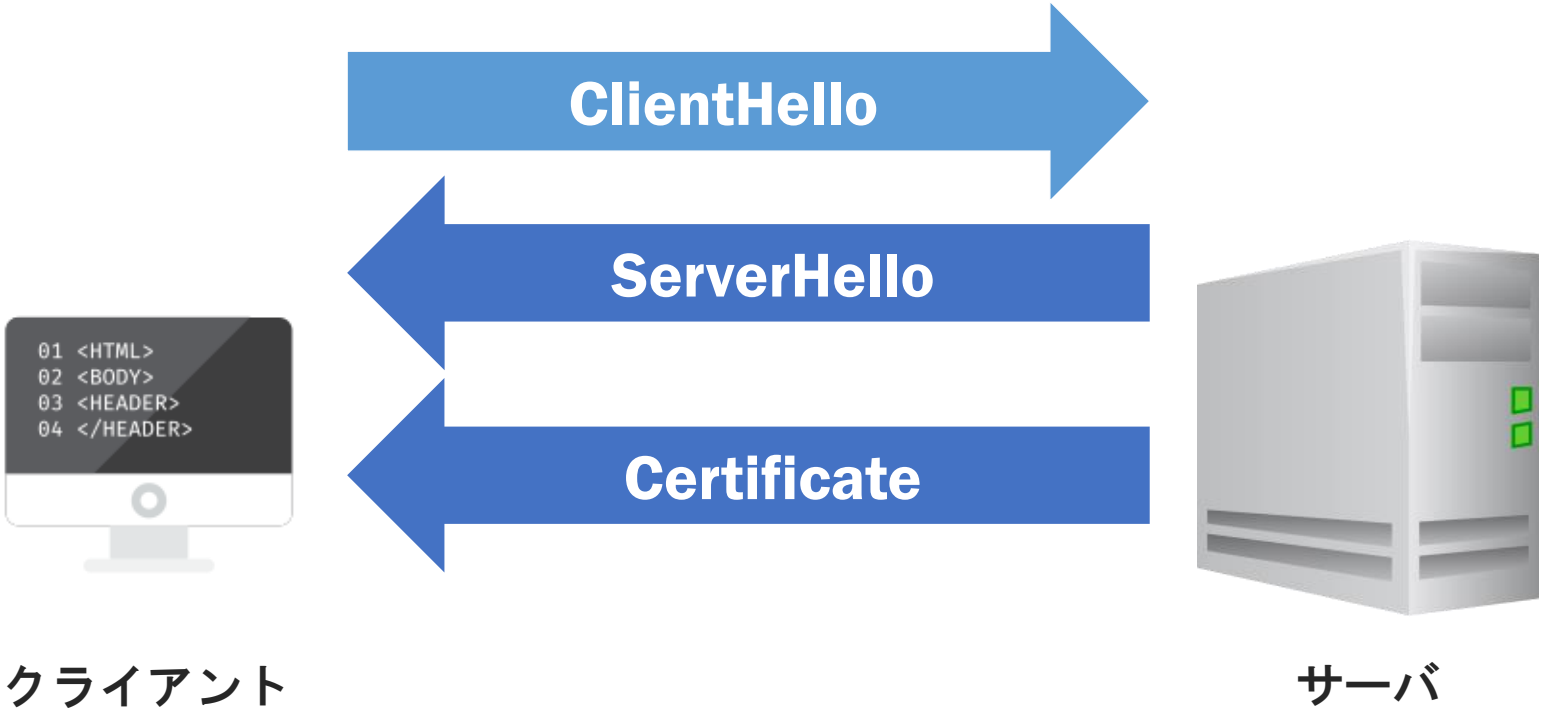
クライアントが対応しているバージョンや暗号スイートを送信

SSL/TLSのハンドシェイク



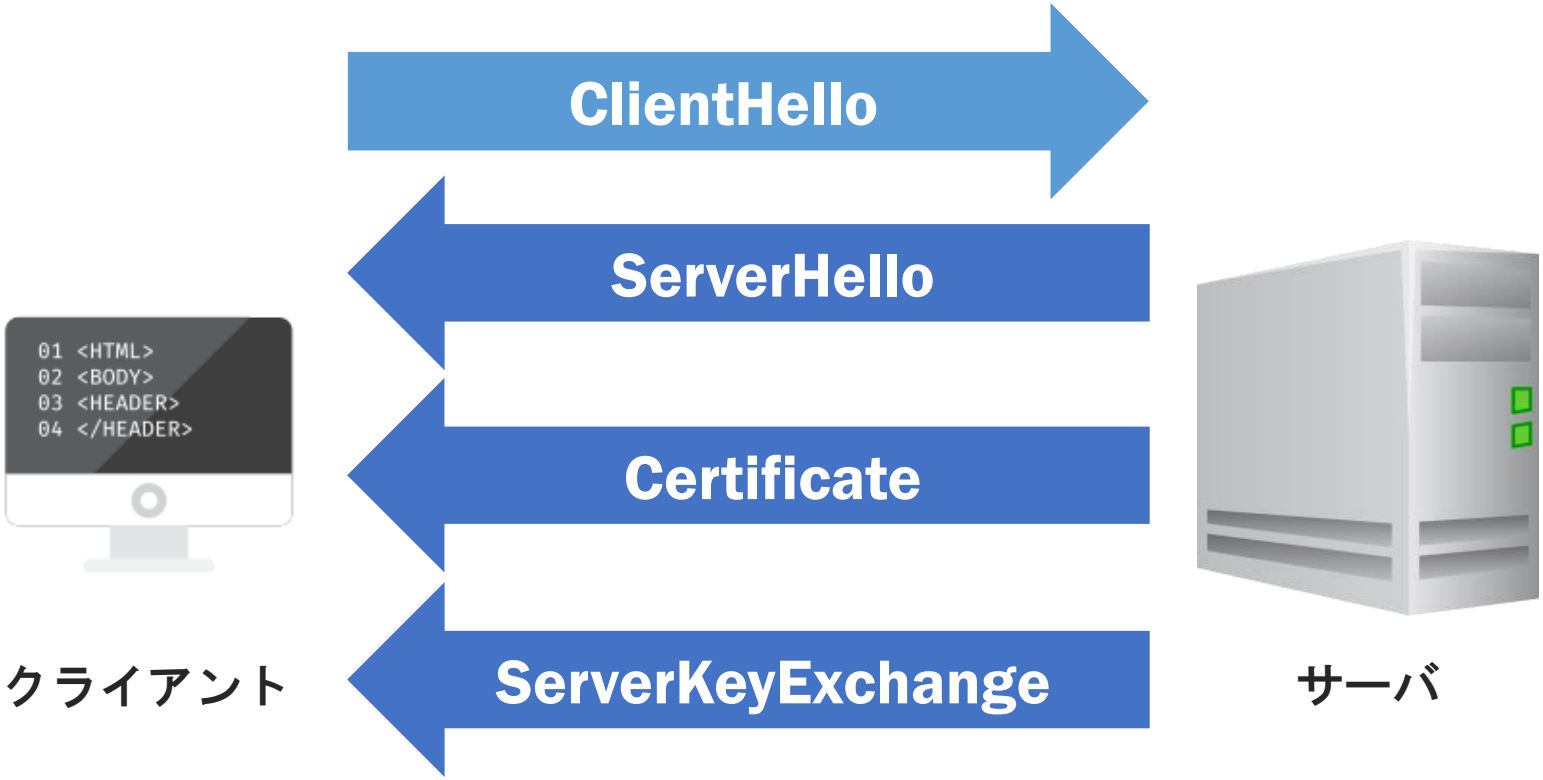
ClientHelloの情報を元に暗号スイートを決定

SSL/TLSのハンドシェイク



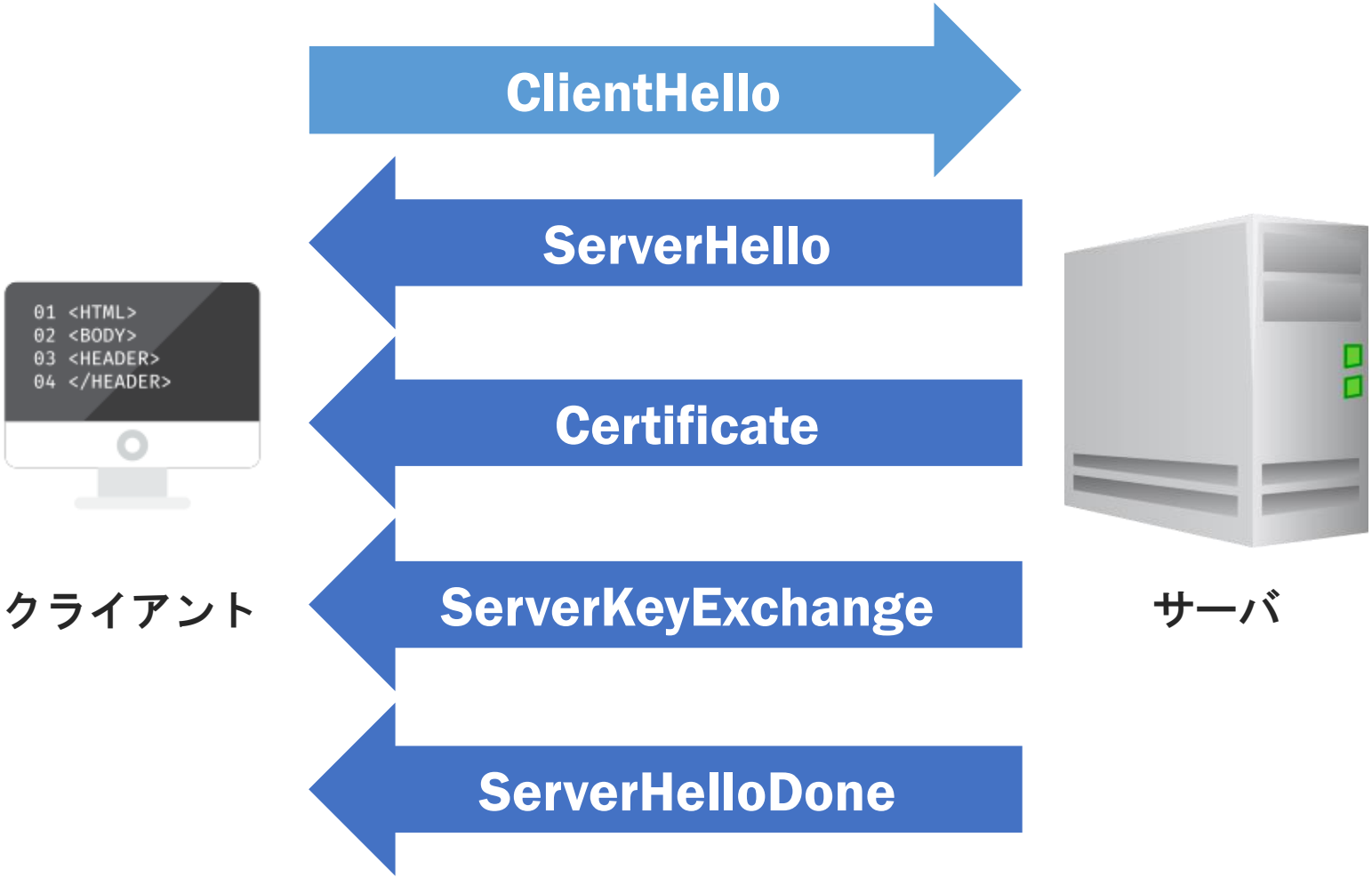
サーバ証明書を送信

SSL/TLSのハンドシェイク



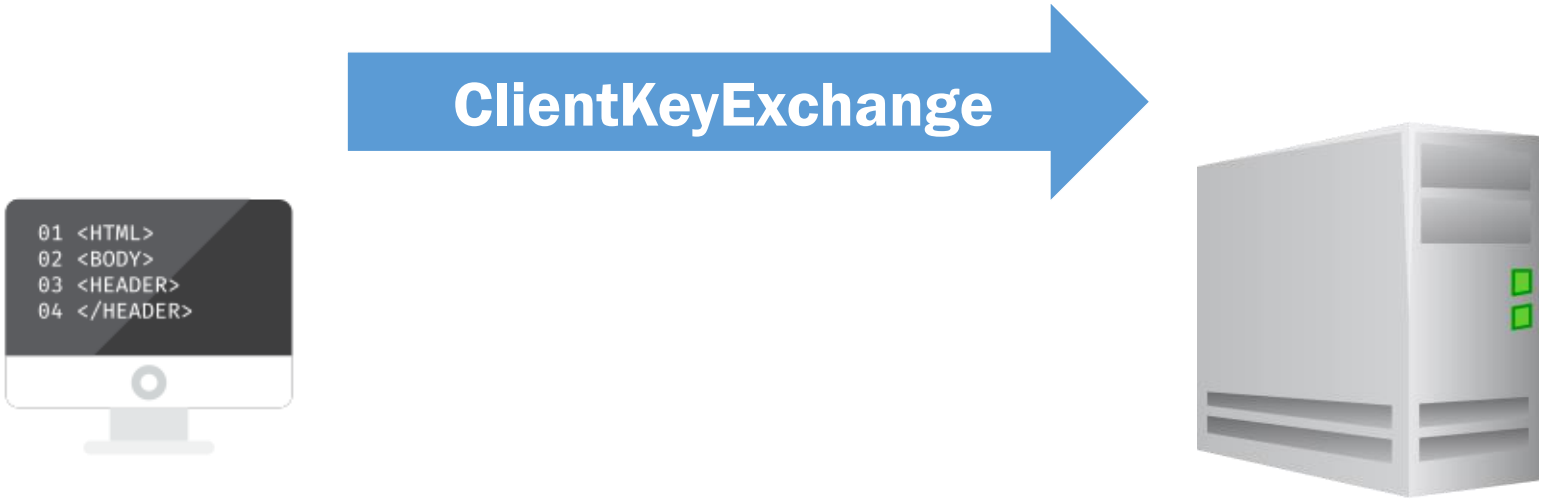
サーバ主導での鍵交換 (無い場合もある)

SSL/TLSのハンドシェイク



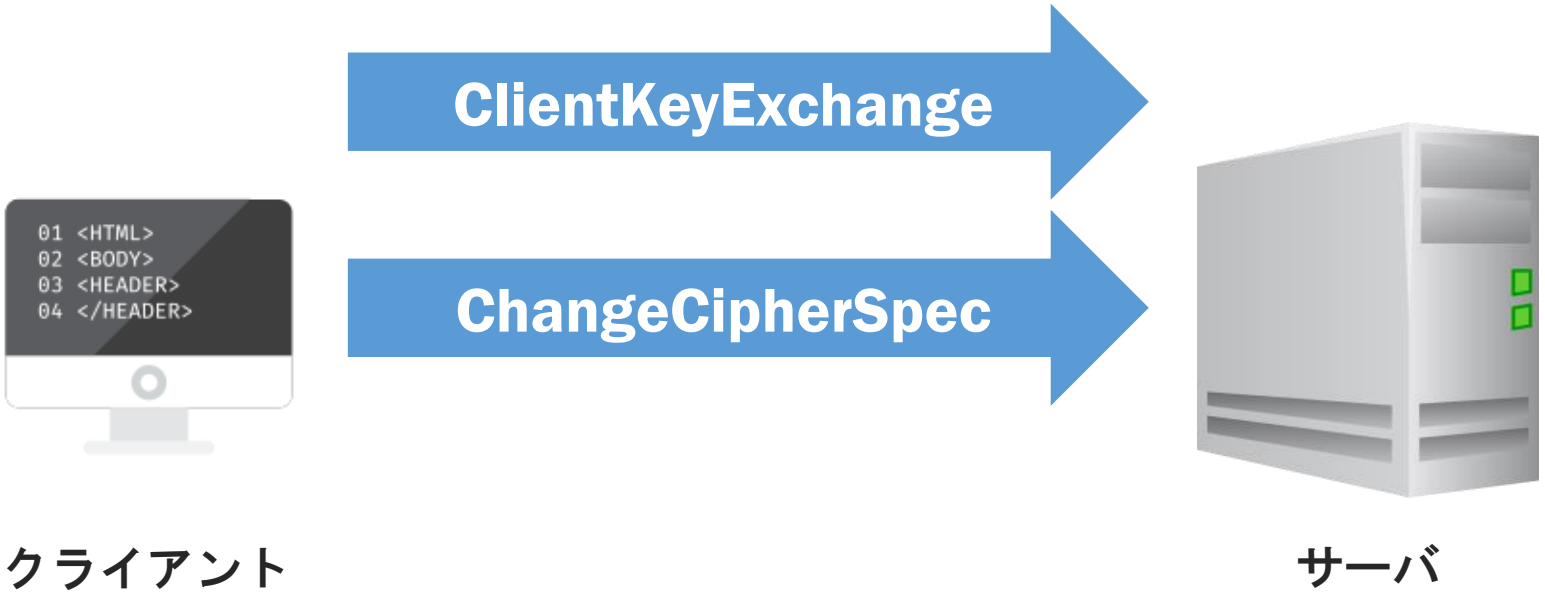
ServerHelloの終了を通知しクライアントを待つ

SSL/TLSのハンドシェイク



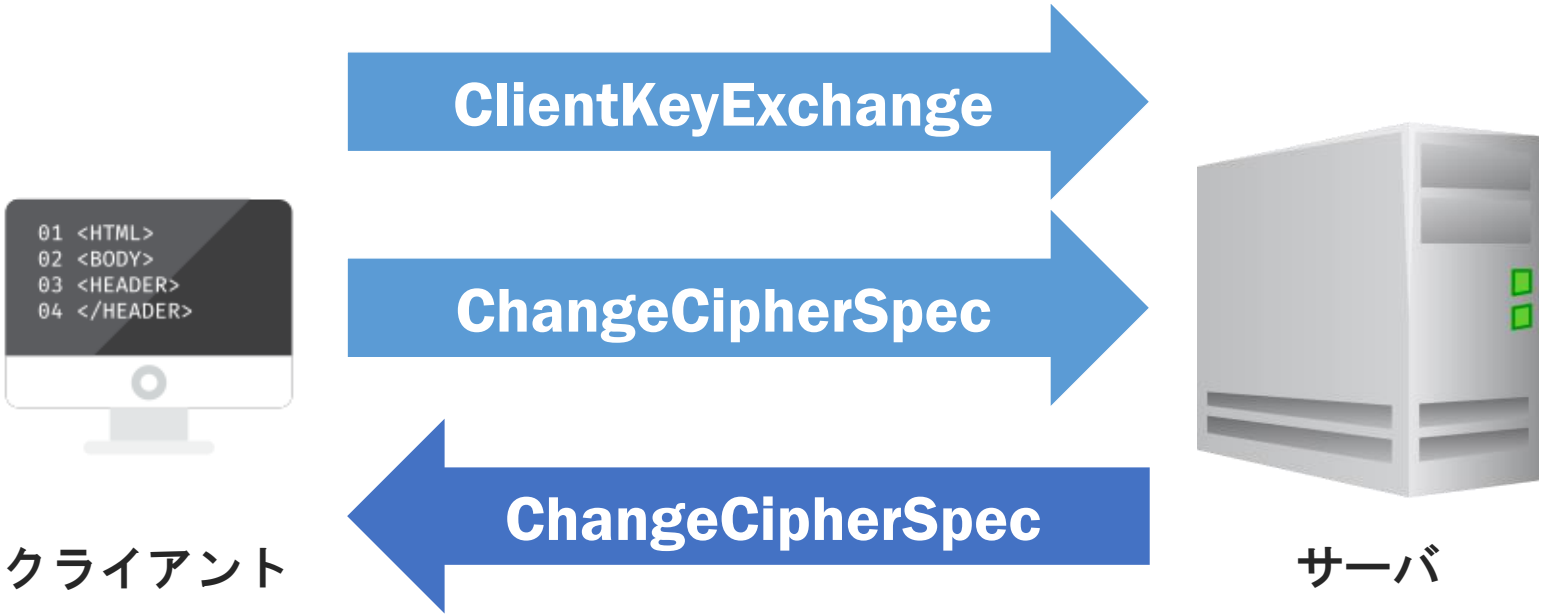
クライアント主導での鍵交換

SSL/TLSのハンドシェイク



クライアント側の暗号化通信を開始

SSL/TLSのハンドシェイク

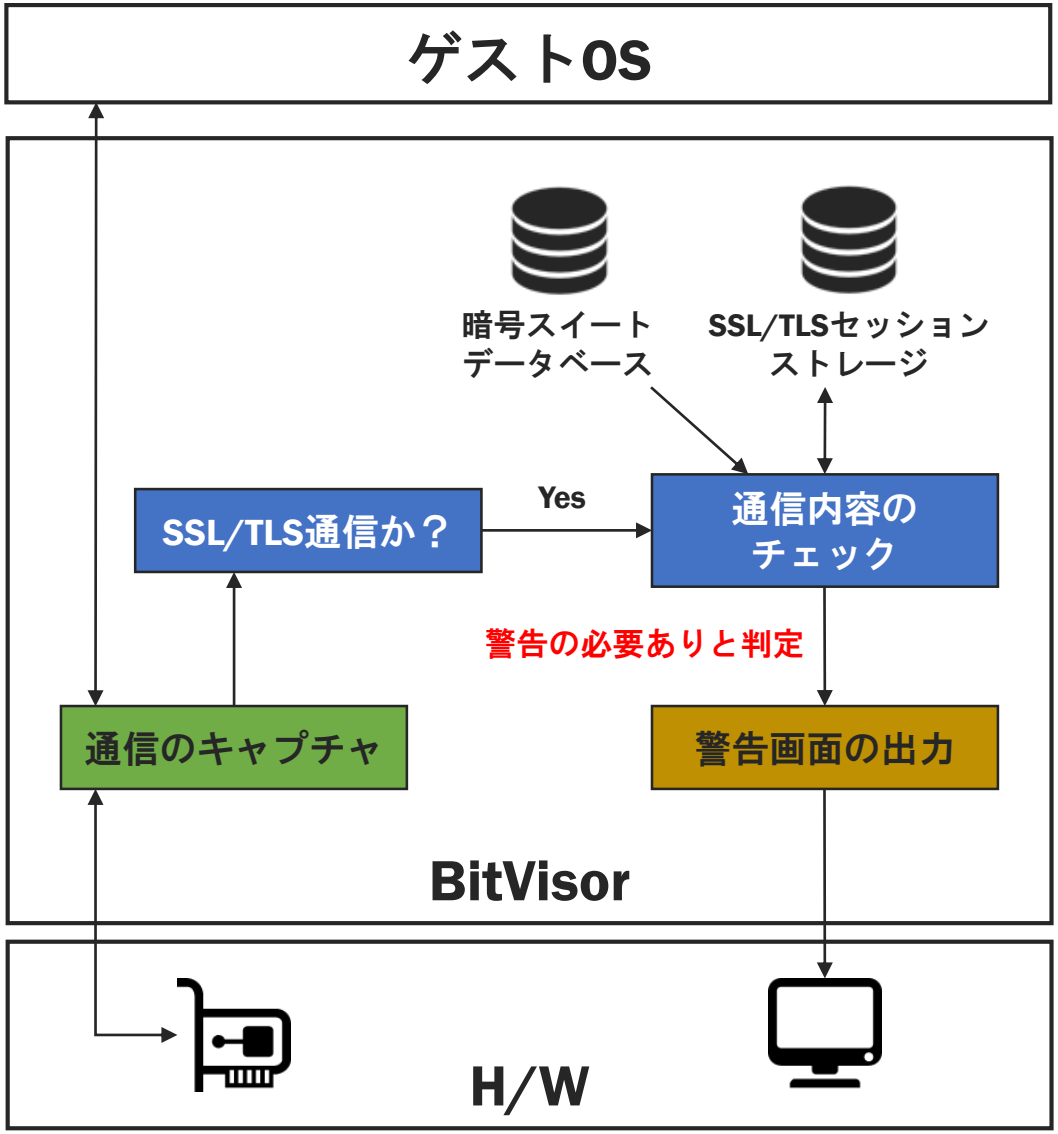


サーバ側の暗号化通信を開始

目次

1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

システム図


ゲストOS
暗号スイート
データベース
SSL/TLSセッション
ストレージ
SSL/TLS通信か？
Yes
通信内容の
チェック
警告の必要ありと判定
通信のキャプチャ
警告画面の出力
BitVisor
H/W

警告の基準

1. クライアントが対応している一番強度の高い暗号スイートのレベルより使用している暗号スイートのレベルが低いとき
   - ▶ レベルとは暗号強度を数値化したもの
     - SSL/TLSライブラリに含まれているものを一部改変
   - ▶ 例外として、RC4については対象としない
2. 弱い鍵交換が行われた可能性のあるとき
   - ▶ 具体的には、鍵長が1024bit未満のとき

SSL/TLSセッションストレージ

- SSL/TLSセッションストレージには各セッションの以下の情報が記録される
  - クライアントが対応している暗号スイート
  - セッションで使用している暗号スイート
- SSL/TLSセッションストレージはBitVisorのメモリ上に存在する

暗号スイートデータベース

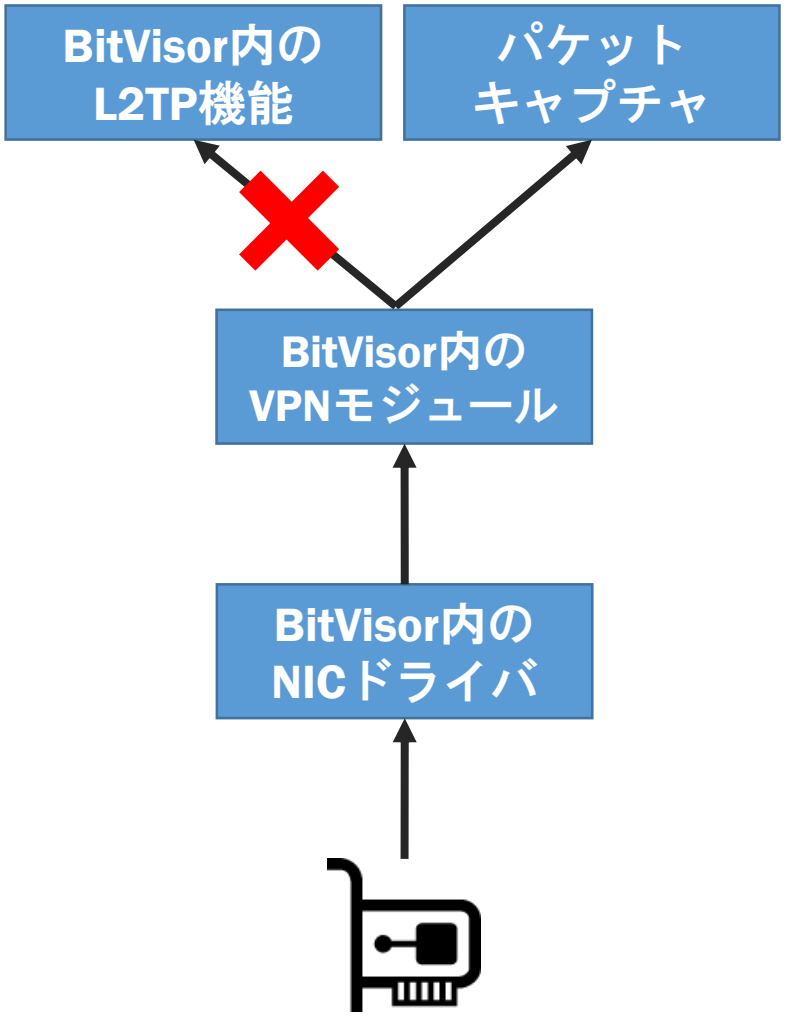
- 暗号スイートデータベースには各暗号スイートの詳細な情報が記録されている
  - 暗号アルゴリズム
  - ハッシュアルゴリズム
  - 強度分類
- 暗号スイートデータベースはBitVisorのイメージ上に含まれている

目次

1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

パケットキャプチャ

- パケットキャプチャにはBitVisorのVPN機能を使用した
  - レイヤ2透過モードを使用した
    (L2TPは使用しない)
  - 送受信の関数にパケットキャプチャのためのコードを追加した



TCPパケットの再構成

- BitVisorに含まれているlwIPは使用できない
  - ネットワークスタックのライブラリなので単なる再構成には使用できない
- SSL/TLS通信はTCPパケットのため再構成を行う必要がある
  - LinuxのTCP/IPスタックをエミュレートするライブラリであるlibnidsを移植して使用した
- SSL/TLS通信のみを再構成する

libnidsの移植

- ネットワーク関連の必要な構造体等を移植した
  - <netinet/*.h>に含まれるような構造体等が対象
- BitVisorに含まれているOpenSSL移植用のヘルパライブラリ(chelp)を使用した
  - chelpのprintf()はクラッシュするため使用しなかった

SSL/TLSセッションストレージ

- SSL/TLSセッションストレージの実装にはlibnidsのセッション構造体を使用した
  - libnidsのセッション構造体には任意のデータを一緒に持たせることができる
  - libnidsのセッションの初期化/破棄とともにそのデータも一緒に処理することができる

SSL/TLSセッションストレージ

❑ libnidsのセッション構造体 (抜粋)

```
struct tcp_stream
{
  struct tuple4 addr;
  char nids_state;
  struct lurker_node *listeners;
  struct half_stream client;
  struct half_stream server;
(中略)
  void *user;
};
```

暗号スイートデータベース

- 暗号スイートデータベースの実装にはOpenSSLに含まれている暗号スイートの構造体配列を使用した
  - 構造体配列に含まれている情報がそのまま暗号スイートデータベースの実装に使用できるため

暗号スイートデータベース

❑ OpenSSLの暗号スイートの構造体配列 (抜粋)

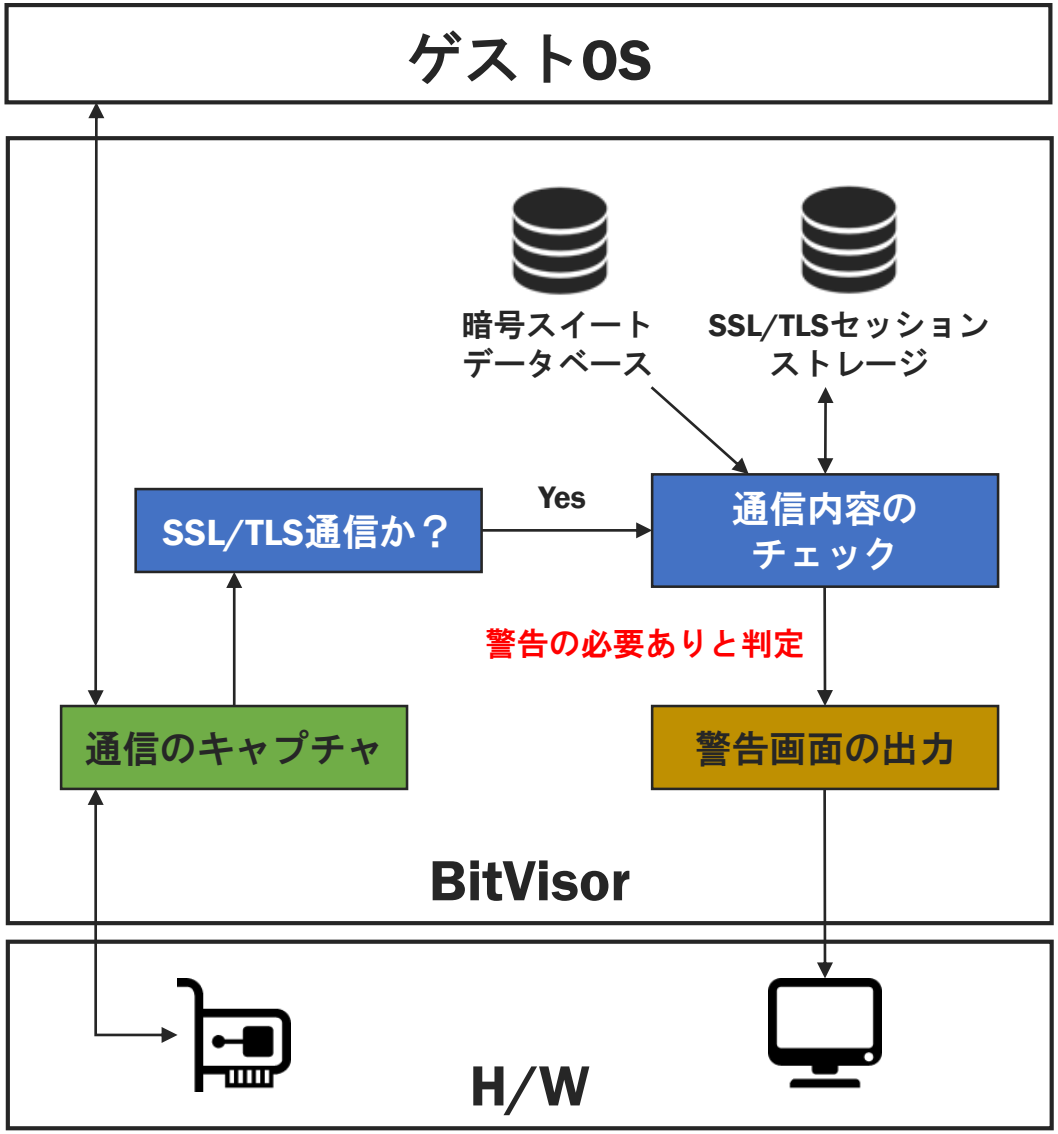
```
const SSL_CIPHER ssl3_ciphers[] = {
...
    {
     1,
     TLS1_TXT_DHE_RSA_WITH_AES_256_SHA,
     TLS1_CK_DHE_RSA_WITH_AES_256_SHA,
     SSL_kDHE,
     SSL_aRSA,
     SSL_AES256,
     SSL_SHA1,
     SSL_TLSV1,
     SSL_NOT_EXP | SSL_HIGH | SSL_FIPS,
     SSL_HANDSHAKE_MAC_DEFAULT | TLS1_PRF,
     256,
     256,
     },
...
```

暗号スイートのレベル

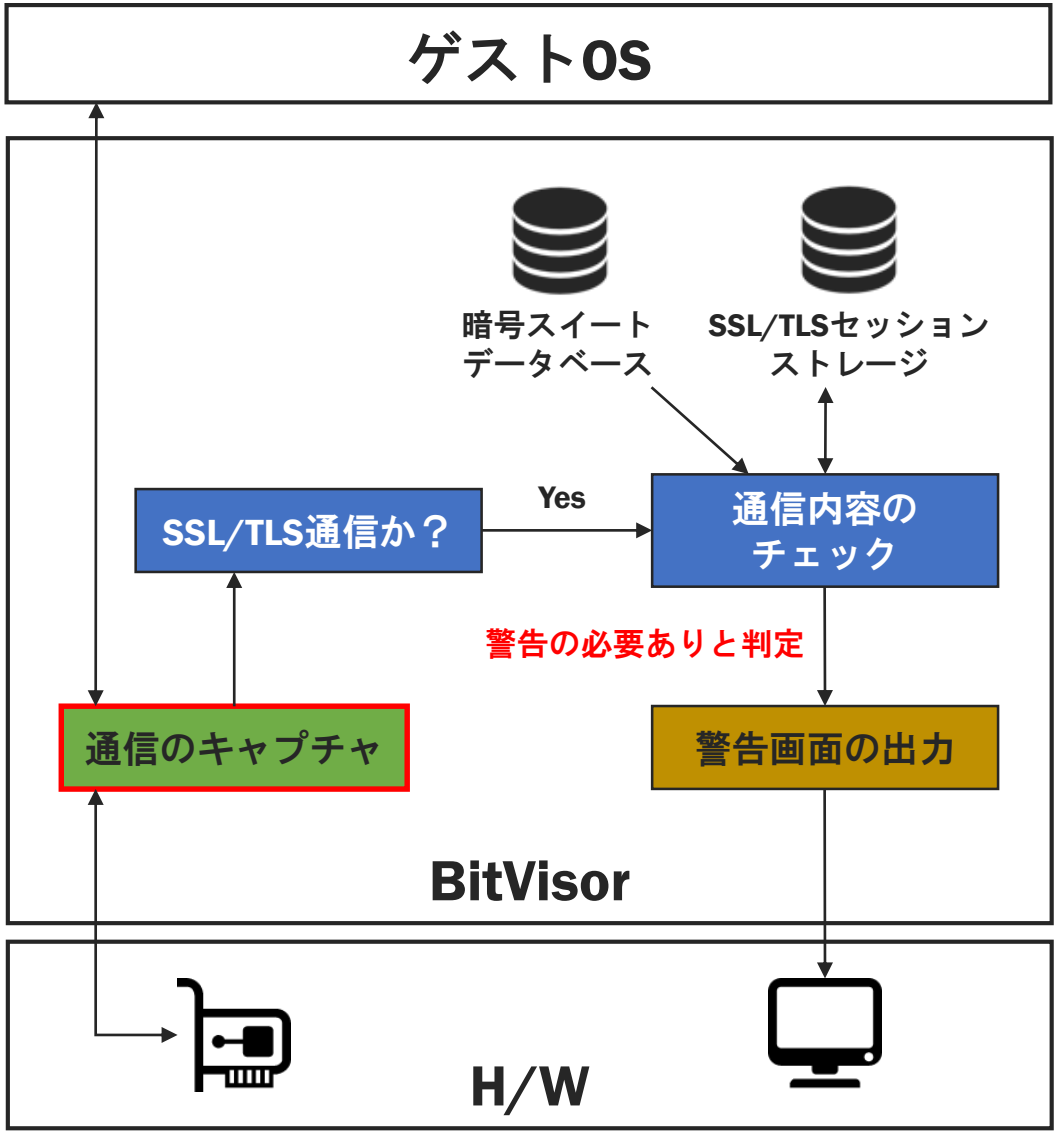
- 基本的に暗号スイートデータベースに含まれている強度の情報から決定される
  - ただし、例外として3DESはレベル3

OpenSSLにおける強度	暗号スイートの例	レベル
NULL	NULL-SHA	0
EXPORT	EXP-DES-CBC-SHA	1
LOW	DES-CBC-SHA	2
MEDIUM	RC4-SHA	3
HIGH	AES256-SHA	4

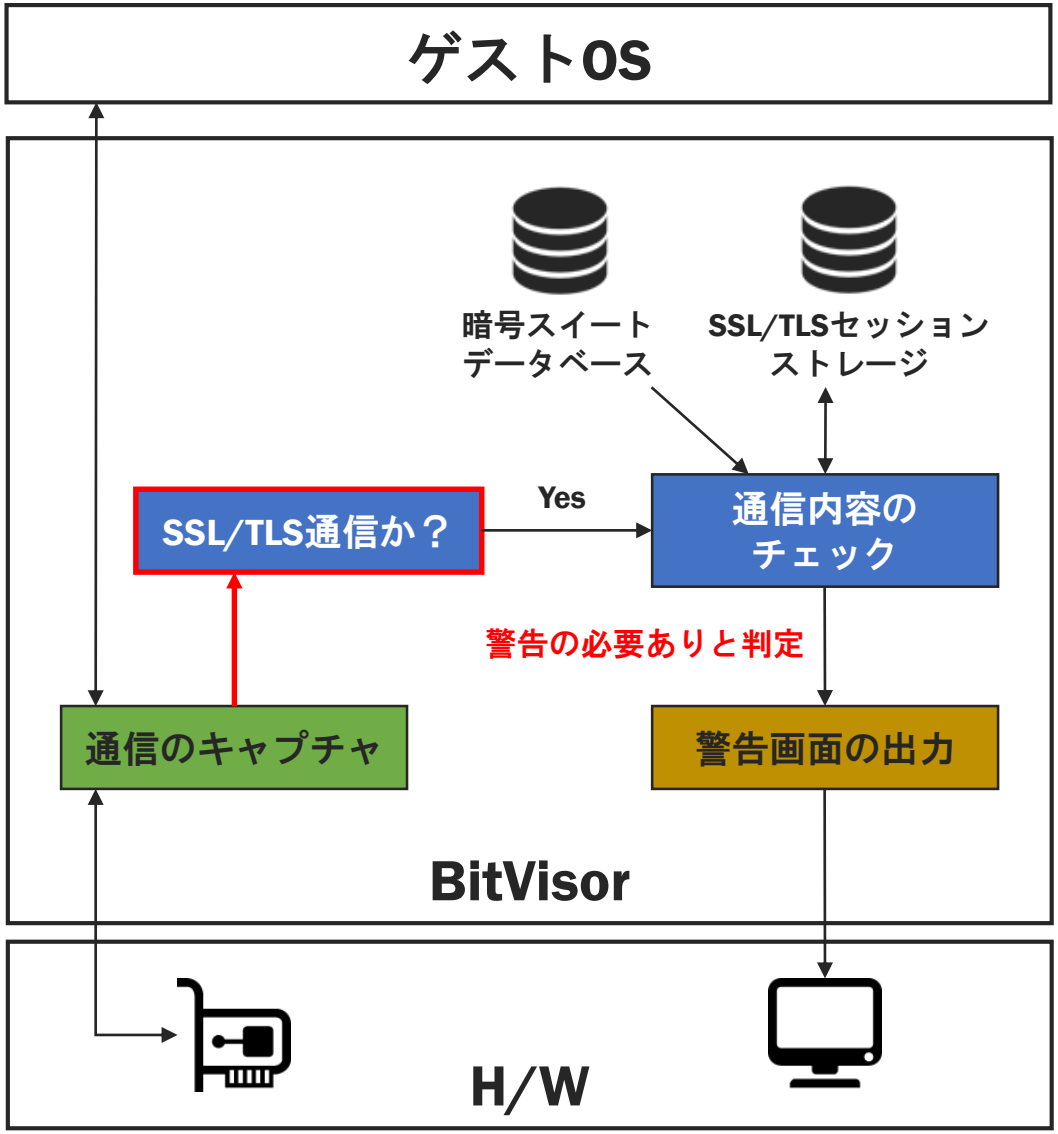
パケットを受け取ったときの動作



パケットを受け取ったときの動作

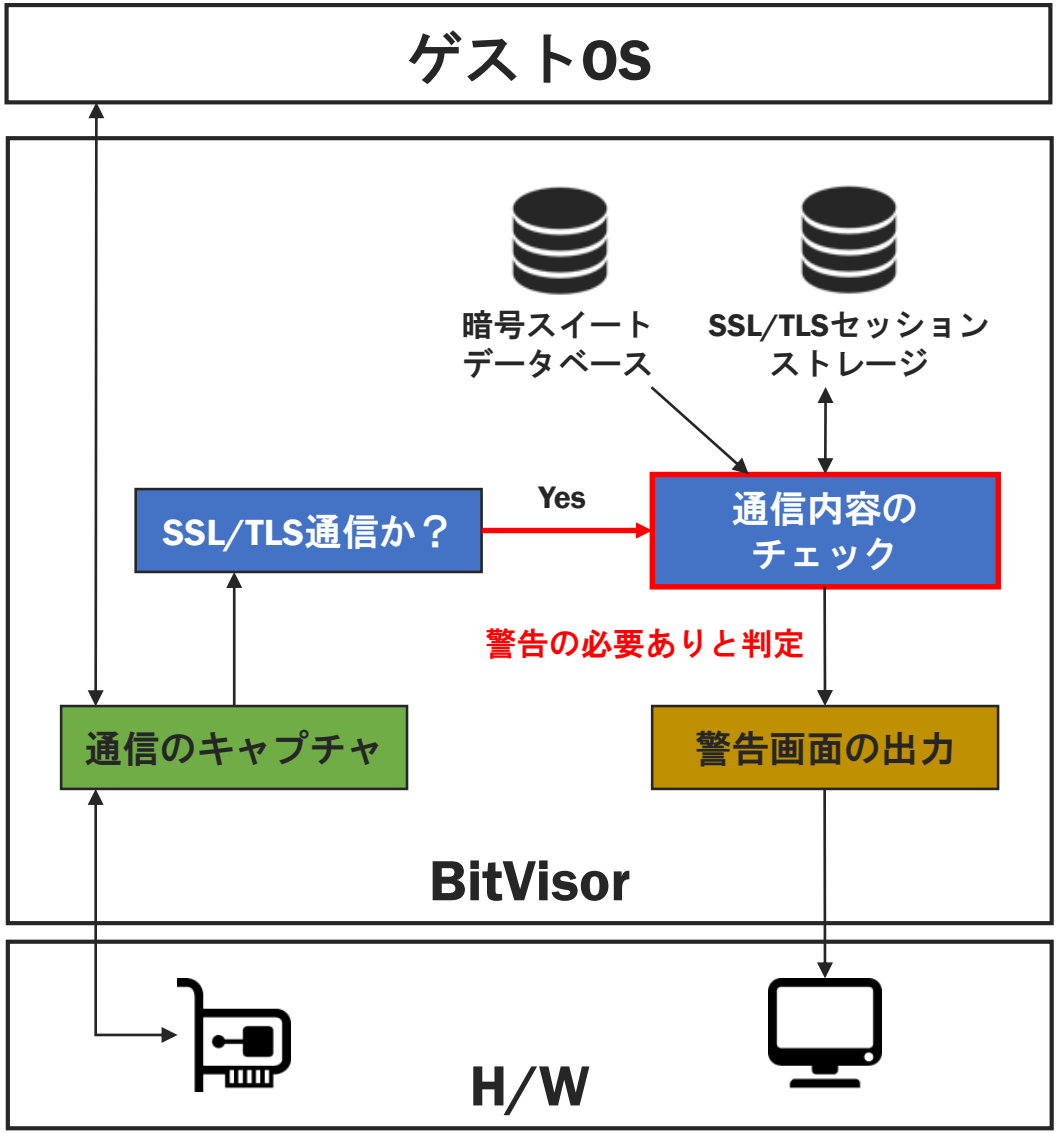


パケットを受け取ったときの動作

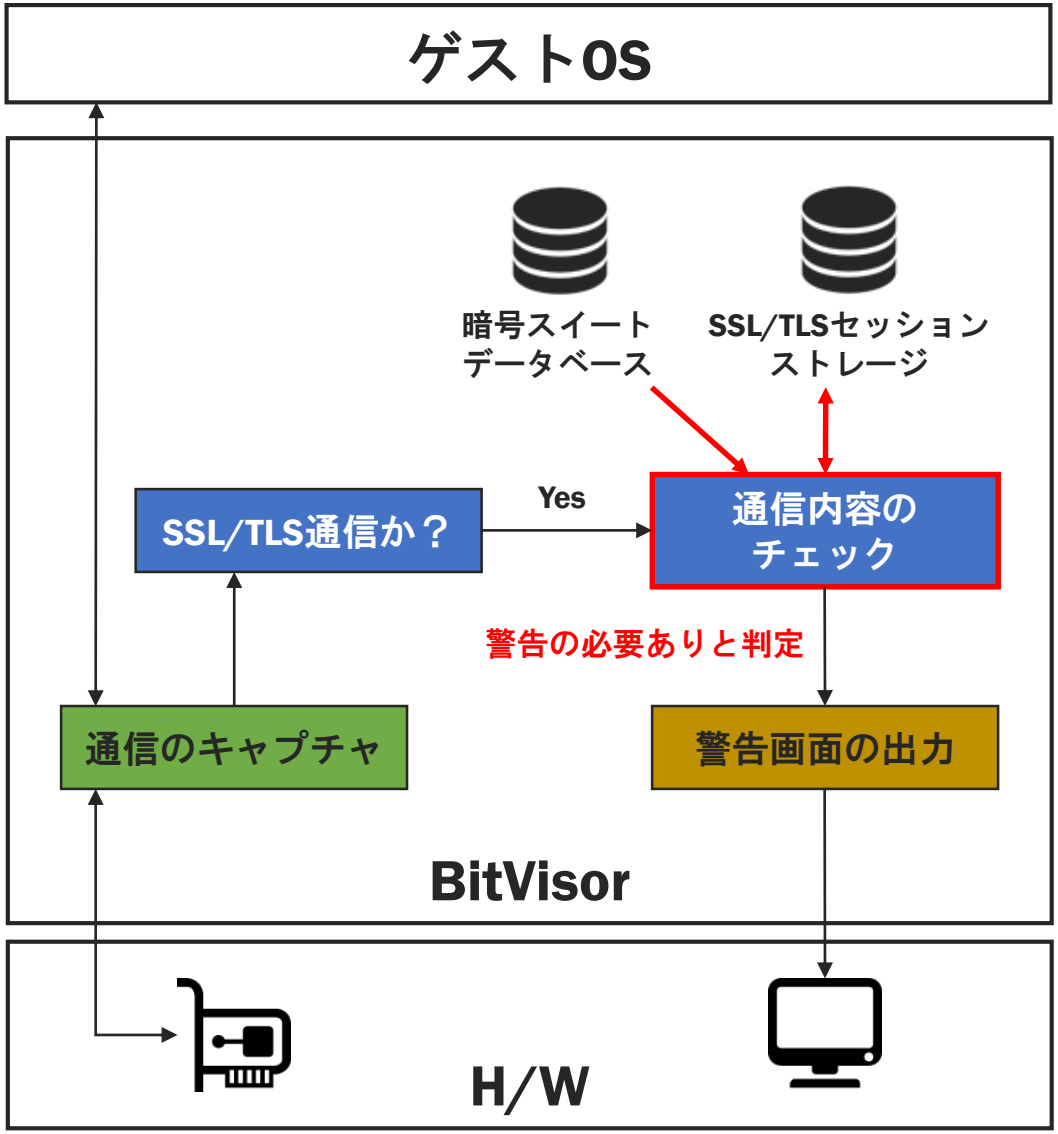


libnidsによって
パケットが処理される

パケットを受け取ったときの動作



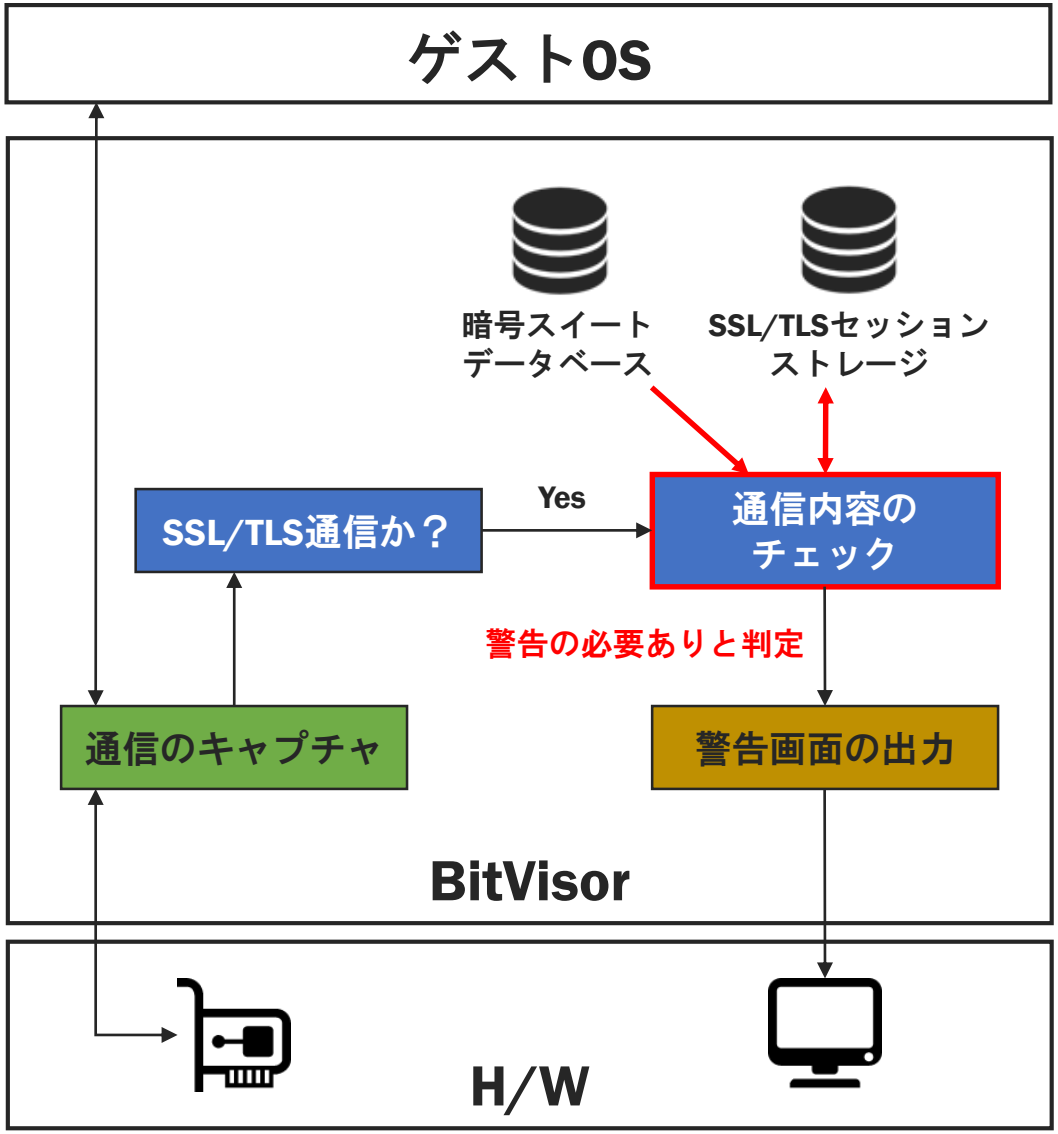
パケットを受け取ったときの動作



ClientHelloを発見時

最も強度の高い暗号スイートをセッションストレージへ保存

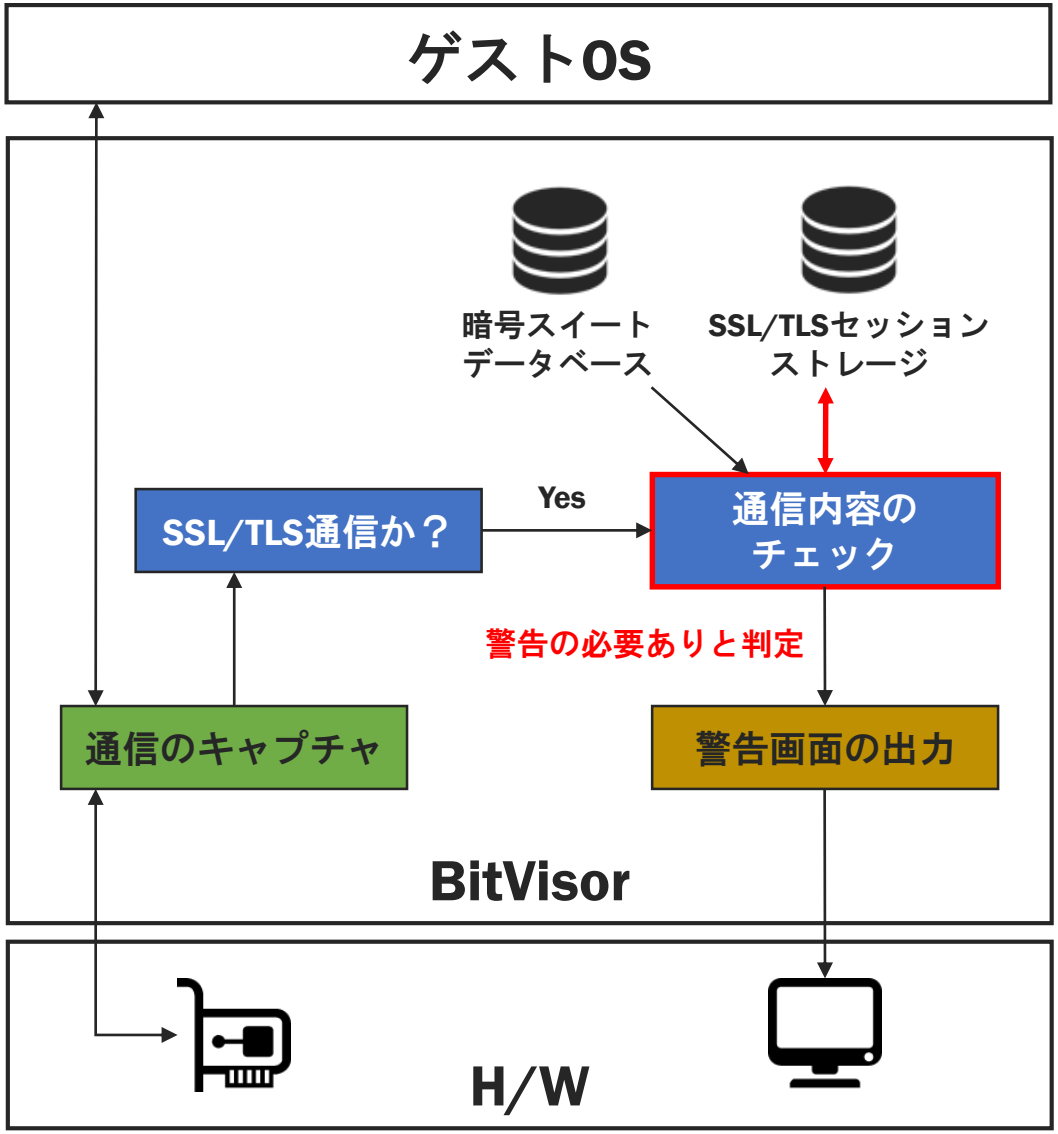
パケットを受け取ったときの動作



ServerHelloを発見時

保存されている暗号スイートと使用されている暗号スイートのレベルを比較

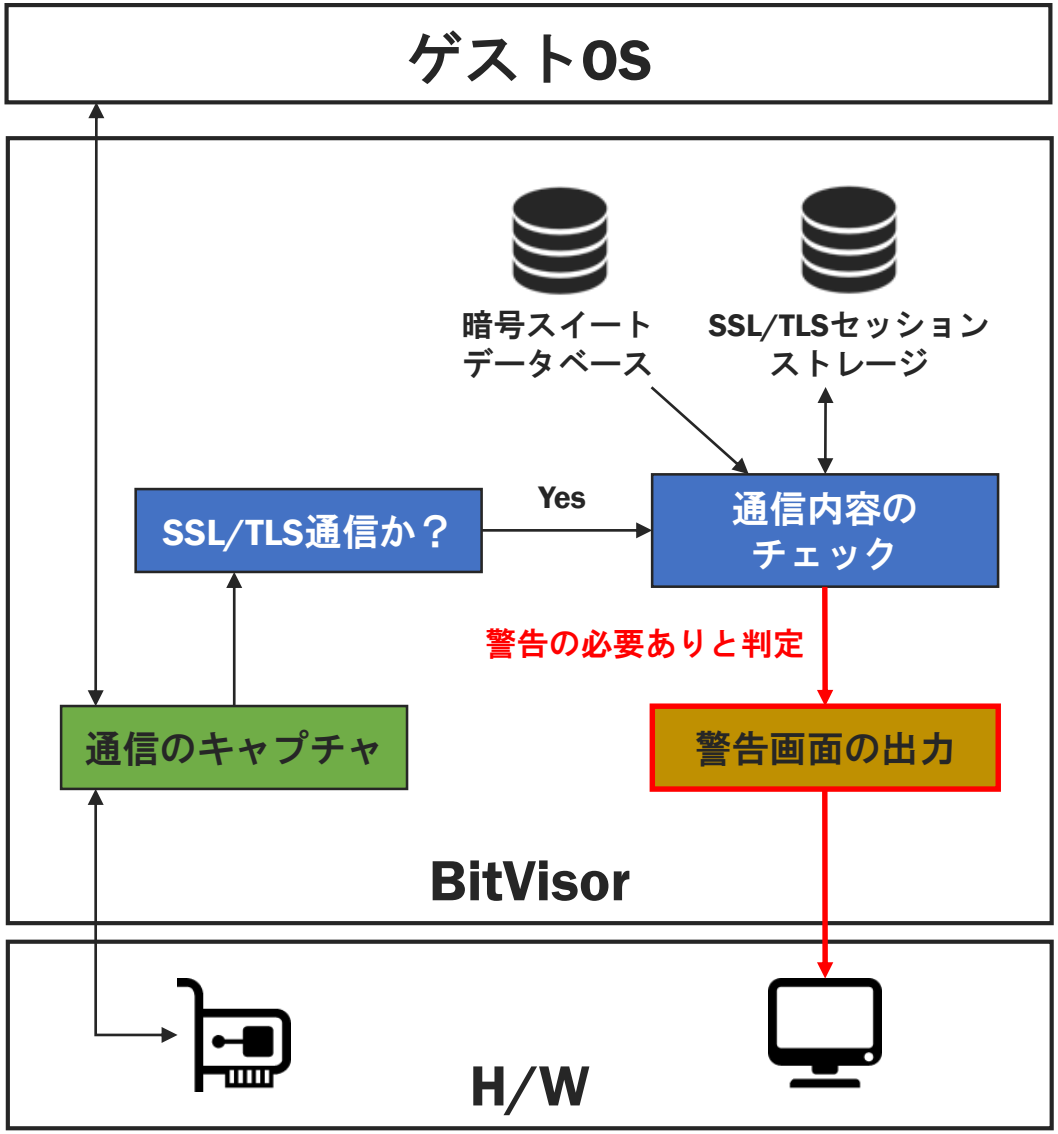
パケットを受け取ったときの動作



鍵交換を発見時

交換されている鍵の長さが短すぎないか調べる

パケットを受け取ったときの動作



必要であれば警告を
行う

警告の表示

- BitVisorに含まれているADvisor機能を使用した
  - ビデオRAMへの直接書き込みによってOSに依存しない画像の表示を実現する
- VPNモジュールから直接使用できないため対策を行った
  - BitVisorに搭載されているメッセージ機能を使用した

BitVisorのメッセージ機能

- msgから始まる各種関数によって、コア機能への間接呼び出しを行う
  - 以下はコード例

```
int desc;

desc = msgopen("vga");
msgsendint(desc, 1);
msgclose(desc);
```

目次

1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

デモ

- https://www.youtube.com/watch?v=C4YPtKyV2pk にあります

1. Debian GNU/Linux 8上
   - ▶ 弱い暗号スイートの検出
   - ▶ 弱い鍵交換の検出
2. Windows 7上
   - ▶ 弱い暗号スイートの検出
   - ▶ 弱い鍵交換の検出

実験環境

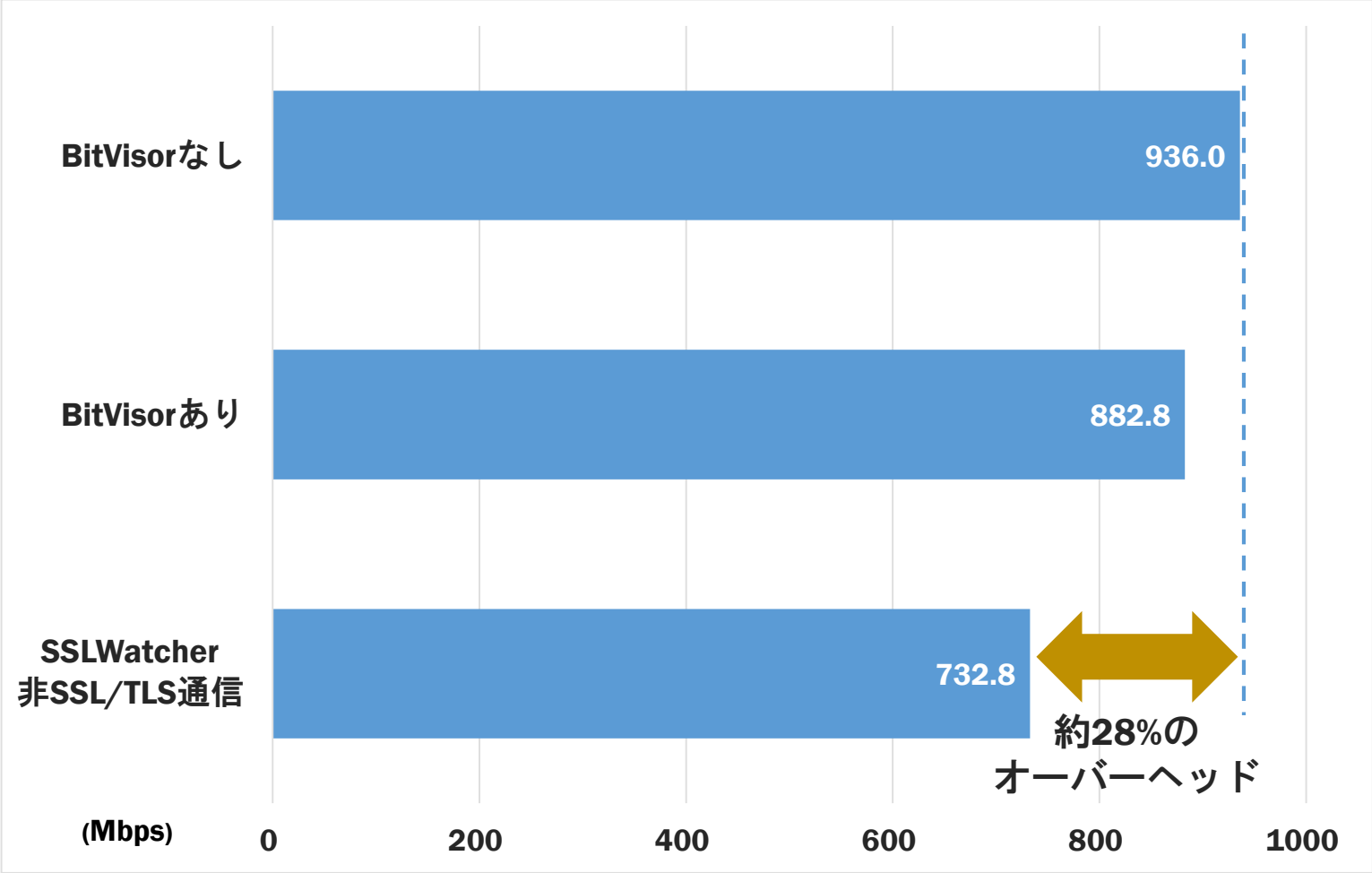
CPU	Intel Core i3 4000M
ゲストOS	Debian GNU/Linux 8
メモリ	4GB
カーネルのバージョン	3.16.0-4-amd64
OpenSSLのバージョン	1.0.1k
BitVisorのバージョン	1.4
ネットワーク	1Gbps Ethernet

通信のオーバーヘッドの測定

- スループットの測定によってオーバーヘッドを測定した
  - 使用したソフトはiperfとwget
  - iperfはデフォルト設定、wgetは1GiBのファイルをダウンロードすることとした
  - 10回測定し、平均値を取って最終的な値とした

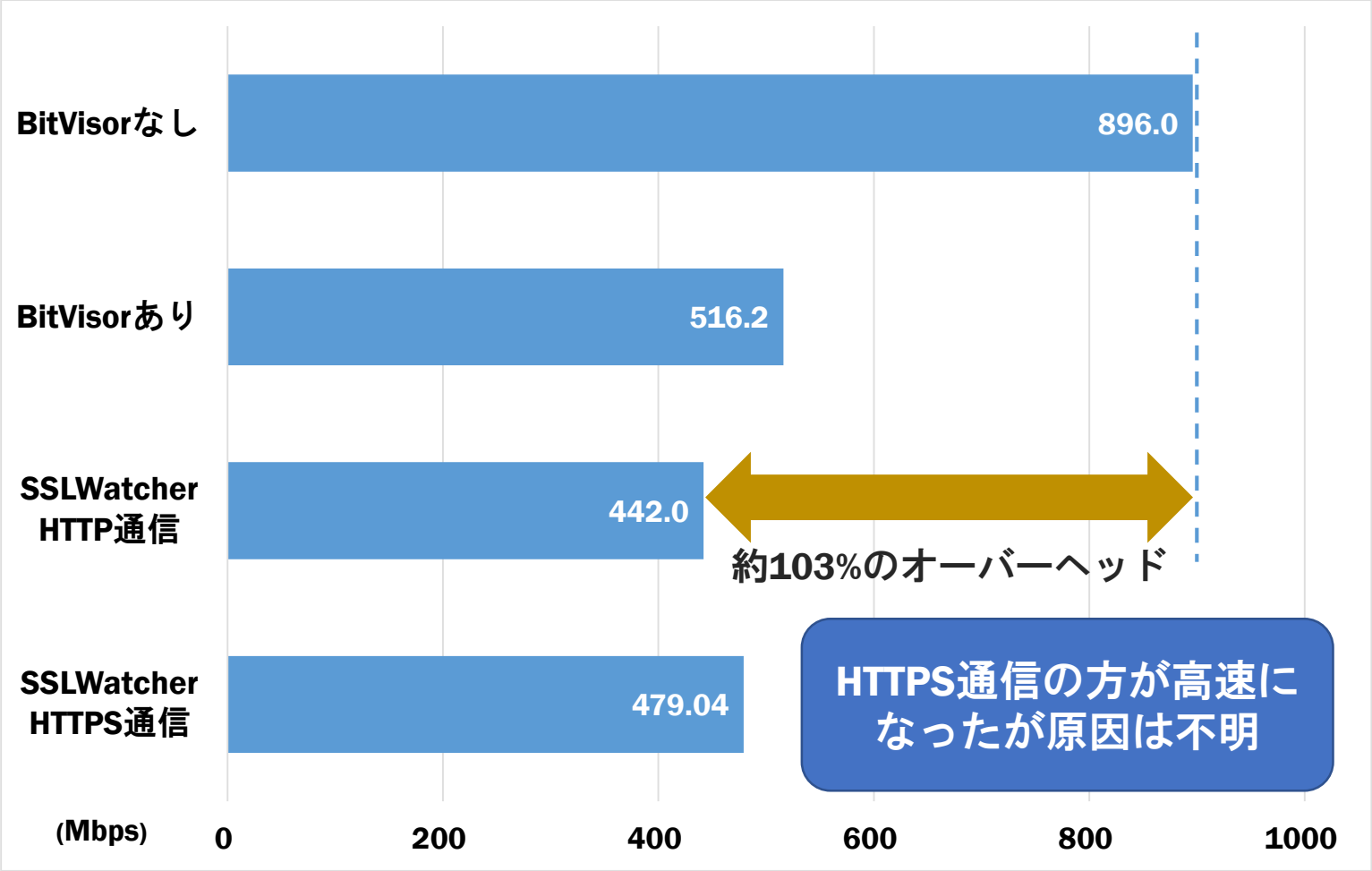
通信のオーバーヘッドの測定

❑ iperfによるスループットの測定結果



通信のオーバーヘッドの測定

❑ wgetによるスループットの測定結果



通信のオーバーヘッドの測定

- スループットの測定の結果、最大オーバーヘッドは以下の通りとなった
  - iperfでは約28%
  - wgetでは約103%
- 実際のユースケースで使用されるプログラムではオーバーヘッドはより少ないと思われる

目次

1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

IDS/IPS

- SnortなどのIDS/IPSはネットワークを監視し不正アクセスを検知するためのシステム
  - どのPCが通信したかをルータを超えて調べることができない
  - SSLWatcherは各PC上に導入されるため、ネットワークの構成に影響されない

SSL/TLSの設定に関する調査

- Leeら [IMC ’07] やLevillainら [ACSAC ’12]
  - インターネット上に存在するSSL/TLSサーバの設定を調査し、セキュリティ的な質を調べている
  - 通信を監視することでSSL/TLSサーバの設定を調査するという方法をSSLWatcherでは使用している

SSL/TLSのセキュリティ基準の決定

- Pukkawannaら [BADGERS ’14]
  - サーバ証明書の情報と暗号スイートの情報からリスクスコアを算出し分類している
  - SSLWatcherはサーバ証明書の情報を使用していないが、ハイパバイザ上に実装されている点などが異なる

SSL/TLSのセキュリティ基準の決定

- MozillaによるSSL/TLSサーバの設定例
  - 互換性レベル別に3段階の設定がある
  - SSLWatcherでは、暗号スイートのレベルの決定や警告基準の例外の決定に使用している

目次

1 研究背景・目的と方針
2 SSL/TLSの概要
3 システムの設計
4 システムの実装
5 デモと実験
6 関連研究
7 まとめと今後の課題

まとめ

- SSL/TLS通信の安全性が低下するような通信を警告するシステムであるSSLWatcherの提案と実装を行った
- SSLWatcher上で弱い暗号スイートや鍵交換を使用するサーバを検出することができた

今後の課題

- 暗号スイートのレベルの計算に使用する情報の追加
  - ハッシュアルゴリズム等
- 検出可能な脆弱性の追加
  - CCS Injection等
- サーバ証明書の情報の使用
- 警告対象の通信の切断
- 通信のオーバーヘッドのさらなる低減