サイバーセキュリティ概論

新潟大学工学部知能情報システム　林隆史

2017/11/20, 12/4

アジェンダ

重要なのは

- 攻撃される可能性を下げること。
- 攻撃に早く気付くこと。

サイバー攻撃から、組織、その活動、ユーザのデータを守る方法について概説します。

- 公開システムと外部アクセスシステム
- 内部システム
- 運用
- 課題と対策

第1部
公開システムと外部アクセスシステム

プロトコル、API、ツールキット、ソフトウェア

脆弱性対策

ログ

ソフトウェアのセキュリティテスト

サイバーセキュリティインシデント

- サイバーセキュリティインシデントは何等かの弱点・脆弱性を利用されることが多い。
  - ソフトウェア、ライブラリ、ハードウェアの不具合
  - アルゴリズムの不具合
  - 設定ミス
  - 不適切な設計のシステム
  - 不適切な運用
- 攻撃を100%防ぐのは不可能
  - 攻撃されたあとの対応
  - 被害の抑制

サイバーセキュリティインシデント対策

- 侵入に使われ得るものを減らす
  - 脆弱性
    - ソフトだけではなく、ハードも
    - Common Vulnerabilities and Exposures (CVE)その他がついているものも、そうでないものも
      - 情報サービスや、各ソフトウェアなどの提供元で随時確認
  - ホスト数
    - プリンター、カメラ、ネットワーク機器も
  - ソフトウェアやライブラリの数
  - ユーザー数
    - 目的に応じて

サイバーセキュリティインシデント対策

- ハードウェアの確認
  - 定期的に、電源OFFと再起動
    - OFFにしたとたん、調子が悪くなることがあるため
  - Criticalな機器は、同一構成も用意しておく
    - ミラーによるコピーをするかどうは、ケースバイケース
  - ハードウェアの脆弱性
    - ファームウェア
    - バグ

サイバーセキュリティインシデント対策

- ネットワーク機器やアプライアンス
  - アプライアンスにはファイルサーバやネットワークストレージも
  - コンピュータ以上にソフト依存
  - ソフトウェアに複数の系列がある場合があり、注意が必要
  - 毎月一回程度、メンテナンス日を設けて、一時休止ができるようにしておく
    - 利用者への周知徹底が必要

ソフトウェアとライブラリ

- ソフトウェアは、様々な機能を利用している。
- 様々な機能をまとめて、「汎用に」使えるようにしたものがライブラリ
- 機能の多様化とともに、ライブラリも、多様化、高機能化、巨大化している。
- 高機能化と巨大化は、一方で、バグやセキュリティ上の脆弱性などを増加させている。

プロトコル、ライブラリの脆弱性

- OpenSSLの脆弱性(CVE-2014-0160(HertBleed))が依然として使われている場合
  - 組み込み系
  - ソフトウェア内臓
  - 独自設定
  - 再起動できてないケース

- **対策**
  - 組み込み系、ソフトで使われているライブラリの確認
    - 特定バージョンに依存しない仕様を作る
      - リソースに余裕を持たせ、ライブラリやソフトの交換可能性を確保する
  - 特定のライブラリをコピーしているソフトのチェック
    - SSLを使っているはずなのに、リンクしていない：　怪しい
  - 重要なライブラリは、代替品を探して準備しておく
  - 入れ替えは通常、簡単ではないので、準備が必要
    - SSLなら、libressl や　mbedSSLなど？
  - ライブラリ入替後は、動いているシステムの確認。

    消し忘れは無いか？。短時間システムを止めてでも、確実に入れ替える。

ツールキットの脆弱性

- **Metadata Anonymization Toolkit**
  - CVE-2017-9149
- Memory Corruption Vulnerability in Foxit PDF Toolkit before 2.1
  - **CVE-2017-7584**
- **Percona Toolkit （Mysql 互換）**
  - CVE-2015-1027 version checking subroutine
- **Strut2**
  - Possible Remote Code Execution attack using REST plugin
  - **CVE-2017-9805**
- EMC
  - CVE-2016-8217
  - An attacker could then feed the modified PKCS#12 file to the toolkit and guess the current MAC one byte at a time.

対策

- ツールキット利用のガイドラインを作る
  - 特定の（新しすぎる）バージョンに依存しない
  - 廃止が予告されたものは、早く取り替える
  - アプリケーションを使う場合には、利用しているライブラリやツールキットのバージョンアップについても確認したり、要求仕様に盛り込む
- ツールキットそのものではなくても、関連したCVEをちゃんと確認して対策をとる
- 利用しているプロトコル（REST, SOAP,MQ-TT）やQuery Language（Graph-QLなど）もよく確認

ライブラリの脆弱性

- 危険なのは、明示的な脆弱性だけではない
- CVEなどで指摘されていなくても、古いバージョンには、方式やアルゴリズムの点で脆弱な可能性がある。
- Linux の各種ディストリビューションのパッケージには、ソフトウェアやライブラリのオリジナルバージョンが固定のものがあるので注意が必要
  - パッケージのライブラリは最小に
    - Alpine Linux
  - 必要なライブラリは必要に応じて、独自管理も検討する
- 特定のソフトウェア群（GNUも含めて）に依存しないことが大事

脆弱性対策　～注意すべき点～

- 情報公開、最初の攻撃、攻撃件数増加、攻撃発覚には時間差がある。
  - どの時点から不正アクセスされていたか、注意深く確認すること
  - 該当するシステムだけではなく、外部との通信ログも調査
- 普段使っていない機能に関する部分は見落としがち。
- 「オフにしてるはず」は危険。

  面倒でも、チェックリストを作成し、定期的に確認。
  - プログラミングスタイルや主流なプロトコル、API、ライブラリは目まぐるしく変わる。
- 昔の常識は捨て、新しいものに対応する。
  - 昔の常識で作られたプロトコルやシステムも理解することが大事

プロトコルの脆弱性

- BIND
  - CVE-2017-3140 , CVE-2017-3141
  - Middle: CVE-2017-3140: An error processing RPZ rules can cause named to loop endlessly after handling a query
  - Critical: CVE-2017-3141: Windows service and uninstall paths are not quoted when BIND is installed
- その他、PHPなどをはじめ種々のプロトコル関連

- **対策**
  - **情報収集（リサーチャー）と自組織への影響評価（キュレーター）が大事**
    - **Notification 他も大事**
  - **プロトコル入れ替えのための準備**
    - **代替物の確認**
    - **入れ替えのための休止や、「片肺運転」のProcedure を明確にしておく**
      - **入れ替え判断の基準も**

プロトコルとその他　～注意点～

- Representational State Transfer (**REST**)などの「スタイル」は有用だが、、、
  - 用いているプロトコルなどへの注意が散漫になる可能性がある。
- Graph-QLはQuery Languageだが、RESTと対比して語られている。
- Graph-QLのように、「新しい」スタイルが出てくると、新たな対策が必要
  - クエリをクライアントで作成してサーバに送りそのレスポンスを利用する
  - クライアント側で必要な情報を定義する：　クライアント側の対策
- 最近の多くの「スタイル」では、全般にAPI変更のコストが大きくなってきている
  - API変更のスキーム作りが重要

ログの対策

- 足りなくては意味がない。
  - 大量のログへの対策も必要
- インシデントの分析を事後にする必要があるのなら、２年分くらいは保存。
- 大量のログがバースト状に発生しても動くようにしておくことが必要
  - NVM Express (**NVMe**) などの利用も

ログ

- 何かトラブルがあったときに、それが故障なのか、設定ミスによるトラブルなのか、侵入や破壊などのセキュリティインシデントなのかを確認する必要がある。
- 確認するためには、何が起きていたのか（起きているのか）についての情報が必要
- システムのトラブルにしても、セキュリティインシデントにしても、「これを見ればわかる」というものがあるとは限らないので、システムの状態を反映した情報（ログ）は残しておきたい。
- 残しすぎると、多すぎて困るという点も大事。

ログ　～チェックポイント～

- 攻撃があった場合に、ログによく残るキーワード
  - **eval(base64_decode()) 埋め込み　など**
  - **Proxyログには、様々な証拠が残されている可能性**
    - **RIG EK（後述）にはいくつかのパターン**
- **攻撃発生時のログには特徴がある**
  - **セキュリティに関連した動作のFail**
    - **”failure,” “failed,” “unable,” “disconnected,” “rejected,”　など**
- 普段ないような通信
  - 特定の外部への通信
  - 特定の内部への通信
    - 内部からネットワーク以外を使って送信されている可能性
- 唐突に表れるログは、何等かの攻撃の可能性も

ログ自体がターゲットになる可能性も。。

- ログに機密情報や残してはいけない情報を記録しているシステムもある
  - マイナンバーをログに入れるようなシステム。（違法の可能性）
  - ログインログに、ユーザが誤って入力したパスワードが入っている場合も。
- ログ自体が重要な保護対象（気密性、完全性、可用性、責任追跡性）
- ログを残すシステムも守らないといけない
  - 可用性を棄損する攻撃は容易
  - ログの保存場所やログの通信路は、必要に応じて多重化

ソフトウェアのセキュリティテスト

- ソフトウェア開発の各段階に盛り込む
  - 最終工程で行うと、最悪の場合設計までさかのぼる
  - 各工程でテストを行い、その結果を他の工程でも共有する
- 日常的にテストを行うことが大事
  - 日常的に行うテストの結果は、設計や構築につながる重要なノウハウ

第2部
内部システム

- 標的型攻撃
- 横展開
- 共有場所
- BIOS他
- Flow

標的型攻撃

- 特定の個人や組織を標的とした攻撃
- ターゲットについて、いろいろと調べた結果を攻撃に利用
  - 普段使っているメールの形式
  - 仕事のフロー
  - 取引相手
  - ターゲットに特有のイベント
  - 使用しているシステムやデータ
- 標的型攻撃を未然に防ぐのは困難
- 標的型攻撃も、最初の攻撃から、情報漏えいなどの重大インシデントの発生までは時間がある。
  - 侵入されたあとに、重大インシデントが起こるまえに対応できるとよい

標的型攻撃　～はじまり～

- 本物と区別のつかないメール
  - メール本体は本物
  - 本物のメールが届いた後、その内容を引用した偽メールが届く。
- 改竄されたウェブサイト経由
  - ちゃんとしたウェブサイトも改竄されている危険性
  - 広告経由(Malvetizing)
    - 広告主のところから攻撃開始。正規の手続きでマルウェアが配信
  - JavaやFlahPlayerを利用
  - 攻撃ツールRIG EK
    - https://www.lac.co.jp/lacwatch/pdf/20170202_cgview_vol3_f001t.pdf

標的型攻撃　～展開～

- 侵入後は、横展開
- 横展開
  - 類似のシステムを探して侵入
  - ユーザ情報や甘い管理者パスワードを探索
  - 侵入台数を拡大
  - 管理者権限の奪取
  - 管理者用システムへの侵入
  - ログの改竄
  - 多数ユーザが使うシステムへのマルウェア
  - 休眠マルウェアやBIOSなどへのマルウェア

横展開の対策

- システム管理者パスワードの管理
  - 共通にしない
- システム管理者のアクセス記録確認
  - 重要なところでは、毎日
- システム管理者についての例外を極力廃止
  - 管理用ホストの特別設定なども
- 不要なホスト間通信の禁止・抑制
- 不要な共有ストレージの廃止
- 外部通信可能なホストやサーバやアプライアンスが無いか確認
  - 電話、無線通信、別のインターネット接続
- ネットワーク上のデータ・メッセージのFLOWの確認
- 侵入されたホストの特定
  - 休眠マルウェア（複数種類の可能性）もあるので注意が必要
- 侵入されたホストが複数でも、怪しい通信は一本化されているケースも
  - 複数の中継ホストを交代させながら使われることも

共有は危険！！

- 共有ディスクはとても危険なので、極力利用しない
- ファイルの共有は危険なので、避ける（禁止する）
- Wikiなども同様
- 適切なアクセス許可の徹底
- SMBやNFSの設定には注意を払う
  - 認証関係　(Kerberos他）
  - LDAPとActive Directory共存の環境では、パスワードを平文で保存したり送信しているところが無いか確認
- 本体はちゃんと管理していても、バックアップデータの管理が適切でないと盗まれたり改竄・破壊される危険性がある

ランサムウェア

- メール、web、SNSなど
- 標的型攻撃、休眠ウィルス他
- 添付ファイル、なりすましインストールプログラム

- 脆弱性のもととなりうるものを減らす
- データのバックアップ

- ランサムウェアをちらつかせた脅迫

脅迫型 Denial of Service

- DoS (Ddos含む)を行うと脅迫
- 対策：　様々な方法の組合せ
  - 複数の代替サービスを用意
  - 複数の異なる場所のサーバ
  - 異なるシステムを使ったサービス
  - Shortest Path Bridge やTransparent Interconnection of Lots of Linksの利用
  - 広域wireless 802.11 j などの利用

代替サービス

- 重要なサービスでは用意すべき
  - ブロックチェーンなども
- サービスによっては、実装だけではなくアルゴリズムも

BIOSへの攻撃

- 通常のマルウェア検知ソフトではスキャンしない部分へのマルウェアが増えてきている
  - 緊急 BIOS リカバリ機能やBIOS感染検証機能を持ったコンピュータの利用
- ディスクのブート領域なども
- 休眠マルウェア

- 対策は難しいが、予備機を用意するなどが必要
  - 予備機に感染しないように工夫

標的型攻撃で侵入から重大インシデントに至るまでの間、重大インシデントの開始

- 侵入範囲の拡大
- 怪しい通信が行われている（はず）
- 普段は行われていないような通信や、あるはずのない通信が見つかれば、それは一つの手がかりとなりうる。
  - 外部への怪しい通信
  - 特定ホストへの怪しい通信
  - スキャンするような通信（ゆっくりとスキャンすることもある）

ソフトウェアとライブラリ

- 使い慣れたものでも、よりよいものが出てきたら置き換えを検討する
- 幅広く動向を調べることが大事
  - CSIRTのリサーチャー
  - 単独では困難なので、連携して集めることと
  - 集めたものを共有し、検索などができるようにしておく
    - グラフデータベースなど
- 調べた内容から、自組織に必要なものを選びとることが大事
  - キュレーター
  - 組織間連携には、工夫が必要

FLOW

- ネットワーク上のデータの流れを調べる
- アプリケーションについても可視化できる
- ファイアウォールやセキュリティ製品による可視化よりも、ネットワーク全体と（アプリケーション）ソフトウェアの利用の最適化と合わせた方がよい
- セキュリティは、機密性だけでない
- 全体パフォーマンスの向上とセキュリティ確保を一緒に行う

第3部
運用

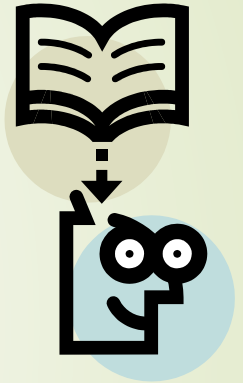
- サービスから実装まで
- プロジェクトマネジメント
- 運用を誰がどう行うべきか

運用とセキュリティ

- 普段のセキュリティを支えているのは運用
- 運用の出来不出来で、セキュリティのレベルは大きく変わる
- セキュリティとシステム運用は、共通するところも大きい
  - 課題解決、課題管理
  - 未解決課題の扱い
  - 計画

セキュリティポリシーと実施手順

- ポリシーと実施手順の区別が大事
- ポリシーと実施手順によって、何をどう守るかを意識
- インシデントが発生しても迷わないように
- エスカレーションや周知をスムーズに
- ポリシーによって、優先順位を明確に



サービスから実装まで

- セキュリティはサービスから実装までに関連する。
- 業務計画、組織ビジョンも反映しないといけない
- ザックマンフレームワークなどを用いた多層的管理が必要
- TO BEとAS ISの比較が大事
  - 日々のチェックが大事

情報セキュリティ・システム計画
Information Security/System Planning

- 大きな計画：　ゴール
- 小さな計画：　ターゲット

- 大事なこと
  - 動かせない達成日　（締め切り）から逆算して計画をたてる
  - 出戻り（何等かの事情で発生するやり直し）を十分盛り込んでおく
  - 課題管理の方法を徹底しておく
    - 課題管理表の作り方
    - 残存課題の管理方法
  - マイルストーンの設定
    - PDCAによるマイルストーン達成の進捗確認と計画修正

計画をたてて実行するには

- 目標と現状との比較が大事
- 目標と現状の比較がちゃんとできれば、
  - 計画や計画を実行するための作業の検討もうまくいく。
  - 計画通りに進まない場合にも、どこが進んでいないのかがわかりやすい
    - 計画の進捗状況を確認しやすいように、計画をたてるとよい。
- 目標と現状の比較をしっかりするためには、目標と現状の把握が大事

To BeとAs Isの確認

- 様々な機能やモジュールを有するシステムでは、
  工夫しないと、確認が大変
- 確認もれをなくす工夫が、
  情報システムそのものとその運用や利用の質を左右する。
- 確認もれが生じやすいところに何らかの工夫をすると、
  確認もれは減らすことができる。

- 確認もれが生じやすいところとは
  - 自分から遠いものはよく確認できない
    - 日時・時期：　夏に検討すると、秋冬のことはもれやすい
    - 自分と縁遠い部署に関係したところ
    - 立場（Zachman Framework の行）が異なるところ

To Be と As Is

- 複数の視点が必要
- 別の視点のものをならべてもよくわからない
  - 別の視点で調べたものは、わかりにくい
- 複数の視点を統一的に扱うことのできる方法が必要
- それぞれの視点での確認も、良い方法が必要

Framework

- よいFrameworkは、To Be やAs Isの確認の手助けになる。
- 見落としを防ぐことができる。
- **確認結果の点検で一番難しいのは、書かれていないことの確認**
- **確認項目のリストは自分で作らないといけない**
- 確認のための指針があれば、見落としを減らすことができる。

Zachman Framework

- 升目におおよその内容が示されているので目安になる
- 行と行の関係が大事
  - 行間は、視点間の関係を表すので重要
  - 表にない「行間」の扱いが決め手
- 関係が大事
  - Zachman Framework is an ontology
    - Ontology： a description or model of the parts and connections that make up a particular field of knowledge or practice

A Typical Zachman Framework

	Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Scope Model						
Enterprise Model						
System Model						
Technology Model						
Detail Representation, Implementatin, and Procedure						

To BeとAs Isの間

- どのようにTo Be に近づけるかの計画には、「マイルストーン」「里程」の設定が大事
- いついつまでに、何をするかをきめて、そのためには、何をしていかないといけないかをリストアップして、ひとつずつ実現していく
- 実現していく過程で、大事なことは

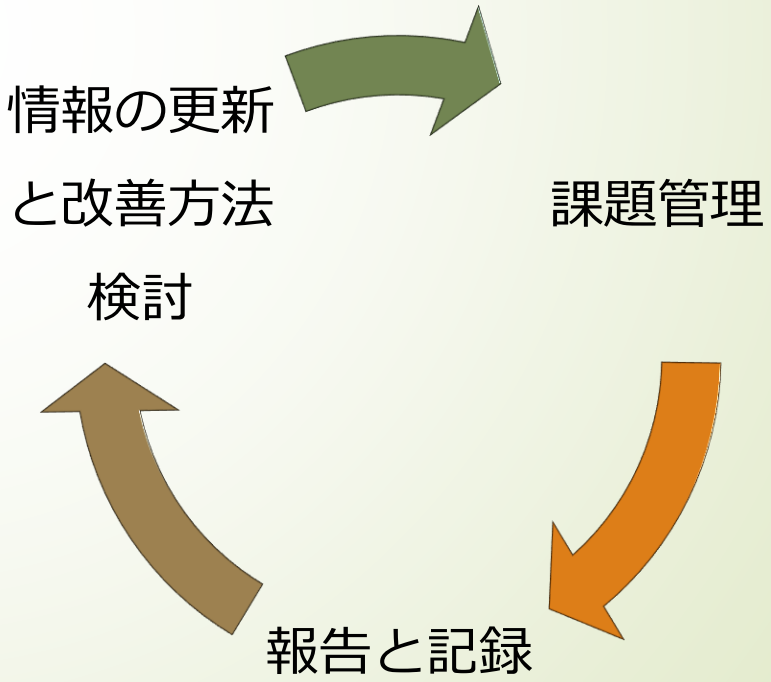
1. Pending、棚上げにした課題のリスト
2. 課題管理
3. マイルストーン設定の修正が必要かどうかの検証
4. 出戻りをあらかじめ想定しておくこと

PDCAサイクル

- 目的を明確にすることが大事
  - PDCAサイクルを回すことは目的ではない
  - Planやその実現方法を現状を的確に把握しながら改善することが大事
- 実際の運用方法については、随時、必要に応じて方法を改善する
- 改善のためには、**記録と報告**

プロジェクトマネジメント

- システムの要件確認・定義、設計、構築、運用とセキュリティの全てにおいて、最重要
- 課題管理
- 報告と記録



第4部
課題と対策

- セキュリティマネジメント
- セキュリティ人材
- 全体安全
- インシデント対応
- 可用性と事業継続性
- 運用

セキュリティとマネジメント

- 情報セキュリティでは、全体安全　Total Optimize Securityが重要
- 多岐にわたるものを見ないといけない
- インシデントが発生すると、短期的対応と中長期的対応の両方が必要
- 他の事例の収集などの調査
  - リサーチャー
  - キュレーター
- インシデント対応時の広報や調整
- 訓練と演習

セキュリティマネジメントで必要なこと

- 課題の完全な終結は難しいことを理解しておく
- 優先順位を常に意識する
- 一見似ていないのに、関連のあるものをうまく扱う
- ワークアラウンドと恒久的対策のバランス

セキュリティマネジメント人材

- プロジェクトマネジメント
- 何が大事かを見失わない
- 一点に注目しすぎない
- いろいろなことを調べることができる

全体安全

- 弱い部分から侵入されたり、壊されたり、盗まれたりする
- 不要な機器間接続を使って、侵入範囲が拡大する
  - 仮想デスクトップ
  - 共有ファイル
- 強力なファイアウォールだけでは守れない
  - 内部のPCへの侵入は様々な方法が存在する。　USBメモリ他
- 検知する機器だけがあっても意味がない
  - 検知したものに対応できる体制が必要

インシデント対応　～短期的対応～

- 関係者への連絡
- 作業の記録開始
- 影響範囲の推定
- 影響範囲拡大の防止
- 類似事例の検索
- ログの保全・退避
- 代替システムの用意
  - クリーンインストール

インシデント対応　～短期的対応～

- システムのフルスキャン
  - 直接関係なさそうなものも含めて全部
    - 検知できた攻撃はダミーかもしれないし、同時に複数の攻撃があったかもしれない
- 重要データの退避と確認
  - 退避する先は、個別に作る（上書きはしない）
- 重要データの確認
- **見かけないファイルがないか確認**
- **サイズが変わったファイルの確認**
- なくなったものがないか確認

インシデント　～事例調査～

- 事例調査をして、自分の組織に関係のありそうなものを記録・整理しておく
- 事例調査をして、取引のある組織に関係のありそうなものを記録・整理しておく
- 関係のなさそうなものも、あったら困るなと感じるものはチェックをしておく
- ある事例について、後に、情報を追加できるようにしておく
  - 後から判明することもあるから
  - 何年もの間、繰り返しおきることがあるから
  - 何年かおきに、類似のでも少し違う事例がおきることがあるから
  - 古い脆弱性も注意が必要だから

インシデント　～レスポンス～

- 戦略策定
  - 定期的に見直すことが大事
  - リスク分析
    - 事例調査
    - 未知の攻撃についても考える
  - 予算
- 情報の分析
  - 脅威評価
    - 全般的評価
    - 自組織への影響度合い
- 対策
  - 事前と事後

インシデント　～レスポンスと広報・調整～

- システムやサービスを止める必要がある場合は、権限のある人にすぐに知らせる。
- 影響のある人にはもれなく知らせる。
- メール、電話、FAX、掲示版などを使う
- わかりやすい言葉
  - どういう影響があるか

可用性
事業継続性

- 障害対策には、障害時の運用に手間がかからないものの方が好ましい
  - Software Defined NetworkよりもShortest Path Bridging (かTRILL)
  - SDNは、障害時のデザインをちゃんとしておかないといけない点に注意が必要。
  - 障害発生時の切り替えのオペレーションが少ない方がよい
- 冗長化は、どこまでを冗長にするか注意深い検討が必要
  - 電源ユニットの一部が原因で冗長化がうまく機能しなかった事例も
  - エッジまでの多重化か、スイッチまでか

Evacuation

- 重要な機能は、サイバーセキュリティインシデント、災害、大規模な故障・障害時に、退避できるようにしておく
- 退避先へのデータや機能のコピー（ミラー）についての注意深い検討が必要
  - 何を対象にするか
  - どういうタイミングでコピーするか
- 機能の切り替え
  - 戻すときも
    - もどしてみたら、何かが無くなっていたということのないように

運用を誰がどう行うべきか

- 製品やサービスの多くは米国製
  - ユーザ側に、多人数の優秀なエンジニアなどがいることが前提
  - 大量のドキュメント
  - 膨大な情報を提供するコミュニティ・サイト
- SIer依存では限界
  - SIerはユーザ側についての知識が不十分
  - 個々のケースについての製品の情報収集の面も対応できない
- どうするべきか？
  - ユーザ側に、セキュリティと運用を行う組織を構築
    - 人材育成（特にプロジェクトマネジメントとプロジェクトデザイン）
    - 短期的には、外部への委託

ご清聴ありがとうございました