セキュアプログラミング（2）
―開発管理―

上原哲太郎
UEHARA Tetsutaro

セキュア開発

- ソフトウェア開発では
  - 機能要件，パフォーマンス要件に加えて，セキュリティ要件

  ・・・　セキュリティ要件は，非機能要件であり，

  その保証についての取り組みは十分ではない

- しかし，もちろん今は，土台となるセキュリティ要件を明確にし，確実に実装することが重要
  - 顧客の信頼
  - 社会的な信用
  - 事業の継続性

    ・・・情報セキュリティ上の被害や法制度の抵触は避けるべき

セキュリティ開発ライフサイクル

- セキュアなソフトウェア開発はプログラマだけの仕事ではない
  - むしろソフトウェア開発という視点で見れば，一部に過ぎない

- ライフサイクル：以下の一連をとらえて考える必要がある
  - 情報システムの企画
  - 上流工程での，適切な分析と定義（設計）
  - 安全なソースコード作成
  - 試験・検査
  - 出荷
  - 出荷後の対応

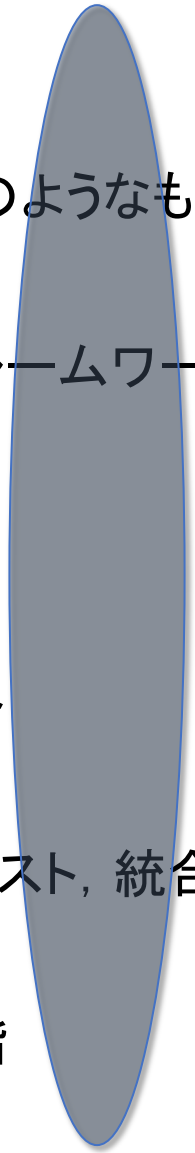
プログラミング以外での検討の必要性

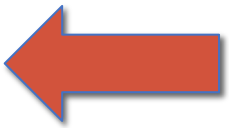
- 要求・設計段階で検討しない場合の弊害の例
  - 情報を攻撃から守る機能が存在しない（設計時に含まれない），または不十分（甘い設計）である
    - インターネットに接続可能な家電
      - デフォルトではパスワードが空で，情報が見放題・・・
- 事後の検討が必要な場合
  - 出荷後にセキュリティホールの存在が明らかになった
  - 社会通念上，セキュリティのとらえ方が変わった
    - 以前は認証があればok
    - 今は認証＋暗号化通信

- 開発段階以外でも十分な検討，事後についても対応ポリシーを決めたり体制を整えることが重要

開発工程

- 要求定義
  - 構築対象システムがどのようなものであらねばならないかを決める段階
- 開発基盤の選定
  - プログラミング言語，フレームワーク，OS，サーバマシンなどを決定
- 設計
  - システムの構造を決定
- 実装
  - コーディング，単体テスト
- テスト
  - テスト計画に基づいたテスト，統合
- 運用
  - システムを運用する段階





セキュリティの検討事項は
横断的に発生する

要求定義

- 利用者が求めている機能・性能の要求を洗い出す
  - この段階で考慮されていない事項を後から追加することは技術的にも，コスト的にも困難
- セキュリティバグの定義
  - 一般のバグとは区別して管理することが推奨
    - セキュリティバグの脅威
    - 原因
    - 優先度

    以上で分類し，記録・修正の有無などの追跡ができるようにする

セキュリティの5大脅威

- STRIDEモデル
  - 脆弱性を有するソフトウェアが，どのような脅威を呼び寄せるか

脅威の種類	内容
Spoofing（なりすまし）	コンピュータに対し，他のユーザを装うこと
Tampering（データの改ざん）	DB等のデータを意図的に操作すること
Repudiation（否認）	ユーザが取った行動を否認し， 相手が行動を証明できないこと
Information Disclosure（情報漏洩）	アクセス権がないユーザに情報が公開されること
Denial of Service（サービス拒否）	サービスの妨害・停止されること
Elevation of Privilege（特権の昇格）	ユーザが，より高い権限を得ること

- 逆に言えば，これらが発生すればセキュリティバグがある

脆弱性の原因

- バッファオーバフロー，アンダーフロー
- 算術エラー
- SQL/スクリプトインジェクション
- ディレクトリトラバーサル　・・・　ディレクトリの公開範囲を超えてしまう
- レースコンディション　・・・　同期・タイミングが原因で起こる問題
- クロスサイトスクリプティング
- 不適切な暗号化
- 不十分な認証
- 不十分な承認・アクセス権
- 不十分な機密保護　・・・　守られるべきものがちゃんと保護されていない
- 無制限のリソース消費
- 不十分・誤ったエラーメッセージ
- 不十分なリソース名・パスの正規化　・・・　名前の解析が不十分
- その他

バグの優先順位

- 対処の優先度を決めてやるべき
  - 被害の規模や範囲，それが発生する可能性など，
    ユーザの要求に従って決める

優先度	定義
1（高）	脆弱性により，致命的な被害を及ぼす可能性があり，その可能性が非常に高い．
2	〃　大規模な被害を及ぼす可能性があり，その可能性が高い．
3	〃　小規模以上の被害を及ぼす可能性があり，その可能性は中程度以上．
4	〃　ごく一部に被害を及ぼす可能性があるが，その可能性は高くはない．
5（低）	〃　すぐに被害を及ぼす可能性があるわけではない．しかし，将来的には予防措置を施すことが望ましい．

セキュリティバグ基準（1）

優先度	STRIDE	内容
1（高）	特権昇格	匿名ユーザが，リモートから任意のコード実行や，与えられた権限以上を取得可能
2	なりすまし	• 別のユーザを装ってサーバに接続可能 • 十分に認証されたユーザと装ってサーバに接続可能
	改ざん	一般的な操作で，本来の手続き・処理を経ずに，ユーザがシステムのデータを永続的に変更可
	漏洩	システム情報，ユーザデータの両方について，公開していない情報が読み取り可能
	DoS	• 少量のデータや短時間の攻撃で容易にDoS攻撃が可能 • 匿名でサービス拒否攻撃が可能
	特権昇格	• 認証ユーザが，リモート・ローカルから任意のコードを実行可 • 〃 与えられた以上の権限を取得可

セキュリティバグ基準(2)

優先度	STRIDE	内容
3	なりすまし	特定または不特定のユーザを装ってサーバに接続可能
	改ざん	特別な操作によって，本来の手続き・処理を経ずに，ユーザがシステムのデータを永続的に変更可
	漏洩	システム情報，ユーザデータの両方について，情報の位置が特定できている場合に限り，非公開情報が読み取り可能
	DoS	• 匿名ユーザが，分散・増幅手法を使わずに，一時的なサービス拒否攻撃が可能 • 認証ユーザが，永続的なサービス拒否攻撃が可能 • 認証ユーザが，分散・増幅手法を使った場合に，一時的なサービス拒否攻撃が可能
優先度	**STRIDE**	**内容**
5	他	• 実際に悪用する手法が知られてないが，セキュリティバグである可能性が高いバグ(曖昧なバグ) • 攻撃の可能性を将来的に減らすことができる設計変更 • いくつかの低リスクの攻撃可能性を緩和するための機能追加

設計

- 定義された要件を基に，詳細な構造を決める．
  - 潜在的な脅威を洗い出して，どのように対処するかを決める
  - 場合によっては，機能の削除や，他の保護機能の追加も判断
- 設計の原則
  - 経済的なメカニズム　・・・　simple is best
  - 規定でフェイルセーフ　・・・　原則「拒否」
  - 完全な調停　・・・　すべてのアクセスについて権限のチェックを
  - デザインの公開　・・・　セキュリティ確保には有効
  - 権限の分離　・・・　1つの状態・条件を前提に許可を与えない
  - 最小限の権限　・・・　必要な処理に，必要な最小の権限を付与
  - 最小限の共有メカニズム　・・・　ファイル・変数の共有は最小限
  - 許容範囲　・・・　セキュリティ優先で「使えない」ものを作らない

（Webアプリの）設計段階で考慮するもの

- アクセス制御に関する設計
  - ユーザ認証をどのように行うか
  - アクセス認可の仕組みの検討
- セッション設計
  - セッション維持やリクエスト強要対策の検討
- ページ遷移の体系
  - 迂回の起こらないように検討
- トランザクション，入力項目
  - ブラウザとサーバ間のデータ項目のやりとり，サーバ側でのデータベース検索や更新のありかた
- モジュール分割

実装段階で考慮するもの

- 昨日や先週したようなことを忠実に実行する

- SQLインジェクション対策
- 入力検査漏れ対策
- スクリプト注入対策
- etc.

検証

- ペネトレーションテスト
  - ブラックボックステストの一種．外部からの攻撃を想定した検証
  - ソフトウェアの全体・一部へ攻撃・侵入を行う．耐DoS攻撃も．
- Fuzzing
  - エラーを含む入力（命令）を自動生成し，無作為に入力する，総当たり的なテスト
  - 人が想像できないようなケースの問題，タイミング依存の問題などを検出することができる
- 動的解析テスト
  - 一定のシナリオ，一定時間動作させ，その状況を監視・分析
- 脅威分析の再評価
  - 設計時の脅威分析を再評価
  - 脅威の漏れの有無，危険度評価の差異の有無

(4) 脆弱性攻撃と攻撃ツールの探索

- 脆弱性の情報は手に入れることができる
  - SecurityFocus (特にBugTraq)
  - Packetstorm
  - Securiteam
- 脆弱性検証ツール
  - Metasploit Framework
  - BackTrack
  - Nessus
  - IBM Internet Scanner Software
- 0-day Exploit: ゼロディ攻撃
  - 未知の脆弱性，未対策の脆弱性に対する攻撃
  - こういった情報を金銭取引するサイトも．

この辺，特に悪用しない．
しかし，
存在は知っておくべき

出荷

- デジタル署名
  - ソフトウェアの出所を明らかにできる
  - その他，オンライン配布の場合に，改ざんの防止にも
- メディアの検証
  - メディア作成の段階でのマルウェア混入を防止
- 流通経路
  - 特にオンライン配布の場合は，配布元サイトの信頼性確認
  - ソフトウェア配布ポリシーの確定

出荷後対応

- 出荷後の脆弱性対応
  - 脆弱性はゼロにはできない．その前提で，誰がどのように対応するかを決定しておく
  - 脆弱性の対応をしない場合は，契約書等に明記

- 脆弱性対応の窓口
  - マニュアル，契約書等での明記
  - ソフトウェア等脆弱性関連情報取り扱い基準（経産省）
    - 発見者
    - 受付機関（IPA，（独）情報処理推進機構）
    - 調整機関（JPCERT/CC）・・・ 開発者への連絡・公表など
    - 開発者