解説

セキュアシステム設計技術
–ディペンダビリティの視点から–

Secure Systems Design-from Dependability Point of View

塚本 克治
（工学院大学情報工学科）

情報社会の名の通り，我々の活動のあらゆる面において情報あるいは情報通信システムがクリティカルな役割を演じつつある．それに伴って，近年，コンピュータ・ウィルスやワーム等に関連して，セキュリティの議論が盛んになっている．しかし，システムの障害はシステムのライフ・サイクルの諸段階での検討不足や過誤による場合も多い．すなわち，システムの障害にもっと広い視野で対処すべきである．そこで，本稿では“ディペンダビリティ（依拠可能性：諸活動をそのシステムにゆだねられること）”の観点を含めセキュリティについて解説する．また，このような知識・スキルを持つ技術者育成のため実施中の「セキュアシステム設計技術者の育成」プロジェクトについて解説する．

まえがき

“脱工業社会”や“情報社会”がD. Bell，梅棹忠夫，増田米治らによって唱えられてから約40年経ち，我々の活動のあらゆる面において情報や情報通信システムは現在の社会でクリティカルな役割を果たしている．一方では，近年，ワーム，サービス妨害（DoS：Denial of Service）攻撃等さまざまな事件が話題になっている．しかし，パソコンのハング・アップや近年の大銀行の統合時のトラブルに象徴されるように，悪意の攻撃がなくても起きる障害は多い．したがって，システム障害をいわゆるセキュリティより広い視野，すなわち，ディペンダビリティの観点からとらえる必要がある．また，開発，設計，運用，保守の現場に即した実践的な知識も重要であるが，問題の複雑化に伴って，対症療法的スキルだけではなく，問題に関する技術者の原理的な知識・理解，たとえば，問題のモデル化や理論的な考察能力が重要になってきている．そこで本稿ではディペンダビリティとセキュリティの概念について解説する．また，工学院大学では，上述のような考え方に立ち，社会人，大学院生を対象に基本知識と実践対応の教育・訓練プログラム「セキュアシステム設計技術者の育成」プロジェクトを進めているので，簡単に紹介する．

ディペンダビリティの歴史と状況

■歴史

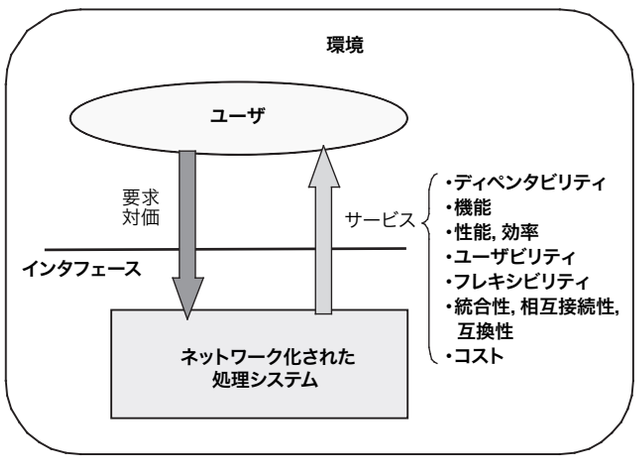
最近の障害事例はマスコミ報道にもしばしば見受けられるので**表-1**には古い事例を主にあげた．障害の解決には新しい理論や技術革新が必要な場合があること，逆に問題が新しい理論や発明のきっかけとなることがある．また，設計や実装での信じられないようなミスが原因になった例も少なからずあること，侵入改ざんはMulticsの例にみるようにその黎明期からコンピュータ・ネットワークとは切り離せない問題であったこと等が分かる．

電子計算機よりも早く登場し，クリティカル・インフラストラクチャとなった電気通信の世界では“いつでも使えること”すなわち可用性（availability）が重視され，

モデルや理論の重要さ，誕生，発明	英–欧大陸間電信の波形崩れ	Kelvinのモデルで説明不可能 Heavisideの電信方程式，Maxwellの方程式の誕生
	電子計算機，通信機器の頻繁な故障	VonNeumannらの信頼性理論誕生
	交換機オペレータの情報横流し	StrowgerのS×S交換機の発明
	データ伝送の誤り	誤り検出訂正符号
設計／実装／操作に起因する障害	Pentiumの浮動小数点の誤り	割り算のアルゴリズムに関するミス
	人工衛星Marinerの失敗	プログラム中のハイフンのつけ忘れ
	Patriotの誤動作（1992年 湾岸戦争）	8時間以内のリブートが必要
	Microsoft DNSエッジルータのダウン	設定ミス
	ARPANET開通時のクラッシュ	（Log in時のG入力でクラッシュ）
侵入・攻撃	MITのTSS優先権の書き換え	（若いスタッフの計算時間割り当ての不満から）
	Texas A&M大学のApple Ⅱのフロッピーのウィルス	プロジェクトのデータ，ID等が破壊される
	Mitinick：たび重なる侵入，1995年逮捕	

■ 表-1　障害事例 ■

一方，情報処理では計算時間の連続性の確保，すなわち，信頼性や計算結果の正しさが問題とされた．1960年代半ばから1970年代には時分割共同利用システム，ARPANET，LAN等新しいシステム形態が生まれた．アプリケーションの面ではバンキング，座席予約等通信と情報処理が融合した実時間の応用が広がった．そして，1990年代にはARPANETから発展したインターネットに経済，社会の諸活動が依存し，システムの設計・実装・保守運用上の誤りやシステムへの攻撃が広範囲かつ深刻な影響を与えるようになった．そして，誤り制御，ハードウェアの信頼性，計算精度等だけでなく，脅威を包括的に理解し，対策を講じることがより重要になり“ディペンダビリティ”の概念が生まれた．



■ 図-1　システムのサービス特性 ■

■ 現状

インターネットの加入者数は世界で6億，我が国で6,000万を超え，セキュリティ確保が困難なワイヤレス製品／サービスの利用も拡大している．Cert/CC[☆1]への脆弱性やインシデント報告の数は，2003年は例外であったが，全体として増加傾向にある．また，CSI/FBI[☆2]の報告では被害額は傾向的に減少しているが，2003年の内訳では，DoSが圧倒的に多く（$26.1M），機密漏洩が2位（$11.4M）である．Synovate社のFTC[☆3]への報告ではID盗用による購買や金銭の被害額は$33Bにのぼり，SANS[☆4]の報告等を総合すると攻撃の傾向は以下のようである：

①セキュリティ製品も攻撃対象の例外でないこと（セキュリティ製品に絶対の信頼を置けないこと）．
②ソフトウェアの供給チェイン，オープンソース・プログラムへのサーバの脆弱性を利用した攻撃があること．
③バッファ・オーバフロー利用が多いこと．
④攻撃の複雑化とそれに要する知識の相対的低下傾向．
⑤攻撃の自動化，間接化，分散化，モジュール化された攻撃手段の組合せ，P2PやIRCによる匿名性利用の増加，複合型攻撃の増加．
⑥整数エラー，タイミング解析の増加．
⑦ Windowsでは IIS，SQL server，IE，Windows Remote Access等への攻撃が多いこと．
⑧UNIXではBINDS/DNS，RPC，Apache Web Server等への攻撃が多いこと．

これらはセキュリティ確保の困難性を示すとともに，これらへの対策が急務であることを示している．

ディペンダビリティとは

“ディペンダビリティ”という言葉は1960年にHosfordが用いていたが，普及は1980年のIFIP WG10.4 “Dependable Computing and Fault Tolerance”の設置，1980年代のLaprieの論文発表以降であろう．Laprieはディペンダビリティを

“ability to avoid failures that are more frequent or more severe, and outage durations that are longer, than is accepted to the user(s)”

と定義している[1)]．ディペンダビリティに関しては信頼性の延長線上の議論も多い．しかし，クリティカルなアプリケーションの拡大，グリッドやユビキタス等の新しい処理形態の登場，開発・設計のグローバル化，オープン・ソースの利用の拡大，手動に頼らざるを得ない分析に対し自動化と回線や処理能力の増大による攻撃の高速化，プライバシーや著作権問題の重要性増大，新しい各種標準の出現等に対応した議論が必要である．本稿では紙数の都合からいくつかの問題に限定して解説する．

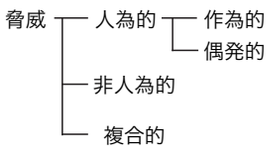
システムのサービスという面からみると，ディペンダビリティは，**図-1**にみるように，機能，性能，ユーザビリティ等とともに提供される価値である．

☆1 米国の国防総省とカーネギーメロン大学がセキュリティ法案にもとづいて1988年に設立したセキュリティ関連の組織．セキュリティに関して情報収集・研究・発表している代表的な機関

☆2 CSI：Computer Security Inst.:セキュリティに関するコンサルタント等を実施する民間団体

☆3 FTC：Federal Trade Commission:米国連邦取引委員会

☆4 SANS：SysAdmin, Audit, Network, Security：セキュリティに関する情報収集・訓練等を実施



■ 図-2　脅威の種類 ■

分類	例
予防	セキュア設計開発, 分割管理, 暗号化 オフライン使用, ファイアウォール
検出・軽減	フェイル・セイフ, フェイル・ソフト
検出・防御／除去	アンチ・ウィルス・ソフト, リコール, パッチ
回復（状態復帰 ＋サービス継続）	自動回復, 自己修復ハードウエア
予測波及防止	リスク解析, 攻撃木解析, アンチ・ウィルス ・ソフト, OSパッチ, 切り離し, 分離
転嫁	損害保険, 損害賠償

■ 表-2　対策 ■

(1) ディペンダビリティの要素：脅威 (threats)，指標特性 (attributes)，対策 (means) をディペンダビリティの要素に挙げることが多いが，ディペンダビリティが確保できないときの損害 (damages) も重要な問題であるので，損害を追加する (脅威とセットとして脅威・損害とみることもできる)．

(2) 脅威 (**図-2**)：脆弱性に対する外部からの攻撃等によりディペンダビリティが侵害される可能性のこと．人為的なものと台風や地震による障害，部品の自然劣化のように非人為的なものがある．人為的なものには作為的 (意図的) なもの，偶発的なもの，複合的なものがある．また，システム資源の改ざんや運用・動作に影響を与える能動的攻撃と情報の盗聴・盗用のように直接影響を与えない受動的攻撃がある．受動的攻撃は検知しにくいことが多い．

(3) 指標特性：ディペンダビリティはシステムの指標特性を列挙することにより定義できる．Laprie らは①セキュリティ (security)，②安全性 (safety)，③信頼性 (reliability)，④保守性 (maintainability) を挙げている．本稿では⑤保守性を保守・検証性に変更し，⑥応答の正確性を加えた．ここで安全性とはシステムがオペレータや環境に悪影響を与えないことである．信頼性はシステムの連続動作時間，言い換えると障害が生起する頻度である．検証不可能なものには依存できないこと，記述や自動的な診断等が重要であるので検証性 (testability) を追加した．

(4) 対策：対策のねらいには **表-2** に示すように予防 (prevention)，検出・軽減 (detection&mitigation)，検出・除去 (detection and avoidance)，回復 (recovery)，損害の転嫁がある．軽減にはシステムのシャットダウンによるフェイル・セーフ，冗長構成によるフェイル・ソフト (graceful degradation) がある．除去の中にはリコールも含まれる．また，ソフトウェア・ベンダによるパッチ・サービスもこれに相当する．汚染された (可能性のある) システムの切り離しも重要である．損害の転嫁の例は保険である．ソフトウェアでは，歴史が浅く，変化が速いので，ハードウェアと一体として扱われる場合を除いて，製造物責任，損害保険 (提供損保はある)，損害賠償の問題の多くは今後の課題となっている．

(5) 損害：損傷・損害の大きさのレベルの設定は，それぞれの組織，プロジェクトのポリシーに依存する．下に示すのは一例である：

①軽微：ハードウェア的な障害はなくテスト・プログラムの検査で検出可．アンチ・ウィルス・プログラムで駆除可．

②小：バックアップ・ファイルからの再ロードまたは再インストールが必要．

③中程度：ハードディスクの破壊，FAT の損傷，ドライブの再フォーマッティング等の攻撃．最新のファイルから回復が必要．

④大：ユーザの使用中に徐々にデータ・ファイルを破壊．ユーザは感染に気づかずに計算機の使用を継続していることがある．検知が困難な場合が多い．一時的に使用不能化．バックアップの信頼性も疑問．

⑤深刻：上に加え，データの破壊等を簡単に調べられない．かなり大きな資産損失，回復費用を要する．

⑥致命的：長期サービス停止．ソフトウェアや設備の破壊，人命の損失等．甚大な経済的，人的被害．

セキュリティ

■ セキュリティ確保の困難さ

(1) インフラの問題 (匿名性と管理) (**表-3**)：電話網に比しインターネットは脆弱性が大きい．ISOC (Internet Society) は差別なきアクセス可能性，検閲の禁止，政府の不介入 (1960 年代後半から 1970 年代の米国社会の雰囲気が投影された ARPANET の性格を引き継いでいる)，すなわち，自由を原則とした．また，インターネットは発信アドレスをユーザが挿入できるこ

	電話網	インターネット
発展の背景	主管庁による規制	ARPANETから発展. 自由
網建設／運用	トップダウン的	ボトムアップ的
参加業者	規制（少数）	比較的自由（多数）
網の階層構造	明確	不明確
管理体系	明確. 一元的	不明確
集中監視／管理機能	あり	なし
網間連携	共通線信号（網）	独立の共通線信号なし
アドレス体系	ロケーション対応	論理的ネットワーク対応
発アドレス	網側で挿入	ユーザ・システムで挿入
加入者データ	大単位（網）で管理	ISPごと（小単位）で管理

■ 表-3　電話網とインターネット ■

と，ネットワークの集中管理がないこと等の問題がある．Ethernetでもセキュリティは考慮外であった．

(2)侵入痕跡消去が容易であること．

(3)人間の介在：人間のミスの完全除去は不可能．

(4) 異常の本質：“異常”を100%定義し，要求定義に盛り込むのは不可能．

(5) セキュリティよりも機能，性能，低コストを優先した開発・設計・運用をしやすいこと．

(6) 既存ソフトウェアの利用：オープン・ソース・コードその他既存ソフトのセキュリティが保証されていないこと．

(7) 請け構造：企業間の連携マネジメントの隙間，企業の総合的な技術力不足．

(8) ツールとノウハウの普及：攻撃に関する知識やツールを利用し，簡単に攻撃可能．

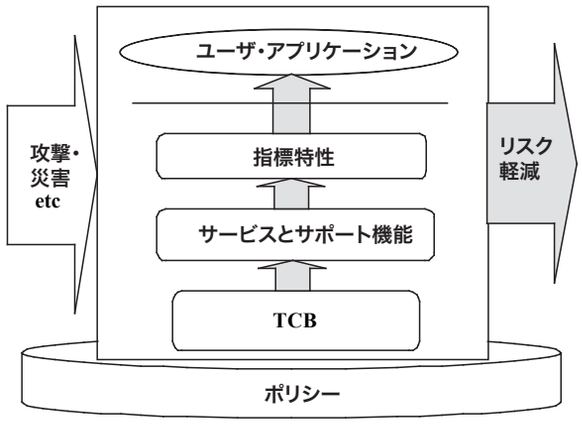
(9) ボーダレス：侵入は国境を超越．国による法制度や体制の相違，捜査権の限界．

(10) 間接的連鎖拡散と拡散速度の向上：多数のホストを踏み台にした間接拡散攻撃の攻撃源追及は困難．回線の高速化，攻撃の自動化，処理速度の向上による短時間大量攻撃に対し，人手に頼る分析・防御では追従困難．

■ サービス・モデル

セキュリティの確保にはモデルをベースとする体系的な考察が必要である．ここではサービス・モデル，システム・ライフ・サイクルとセキュリティ管理サイクルの面から述べる．

図-3はNIST SP800-33を参考にしたセキュリティ・サービス・モデルである[2]．サービス・モデルはユーザからみたセキュリティの指標特性を定義し，それら相互の関係，それを実現するためのサポート機能，TCB（Trusted Computing Base），ポリシーとの関連を論じるモデルである．



■ 図-3　セキュリティ・サービス・モデル ■

可用性（%）	負稼働分／年	グレード
90	52560.00	管理不十分
99	5256.00	一応管理
99.9	526.00	よい管理
99.99	53.00	Fault-tolerant
99.999	5.00	高稼働
99.9999	0.50	超高稼働
99.99999	0.05	超超高稼働

■ 表-4　可用性のレベル ■

(1) 指標特性：セキュリティを定義するシステムのサービス特性である．ISO/IEC17799では完全性（integrity），可用性（availability），機密性（confidentiality）を挙げているが，上記NIST文書では責任遡及性（accountability）と保証（assurance）を追加している．本稿ではさらに近年の状況変化を考え，権利保護を組み入れた．ここできわめて簡単にこれらの意味を述べておこう．

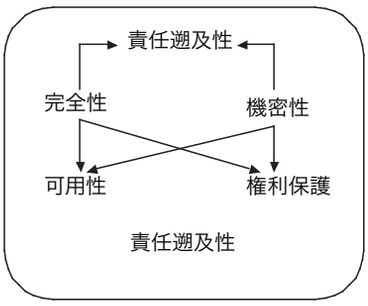
①完全性：不当な改ざんや操作をされないこと．データ完全性とシステム完全性とがある．

②可用性：いつでも正当な要求に応えられること．可用性の指標を**表-4**に示す．電話網は1時間／20年＝3分／年で可用性の設計値は高い．

③機密性：不当に第三者に情報が漏洩したり，システムが外部に不適切にさらされたりしないこと．情報の機密性のレベル設定について種々の案がある．組織等のニーズに合わせ設定すればよい．

④責任遡及性：障害の原因となった根源，責任を追求し明らかにできること．攻撃抑止，障害の切り離し，DoS追跡，法的措置，検出と除去，回復，非否認のための基礎であり，ポリシーの必須条件である．

⑤権利保護：プライバシー保護と知的財産権保護．個



■ 図-4　特性相互の関係 ■

暗号化鍵＼アルゴリズム	公開	非公開
公開	公開鍵暗号	–
非公開	対称鍵暗号	古典暗号

■ 表-5　暗号の種類 ■

認証手段	有益性	弱点	例
知識 (What you know)	実現容易, 携行可能	盗聴, 攻撃発生の検出不能. パスワードは覚え難く, 忘れやすい. 受動的攻撃に弱い. パスワード忘れのコスト, 拾得者に使用される危険.	パスワード, PIN 等
所有物 (What you have)	不正利用困難	高価. 紛失 or 盗難. ハードウエア障害. 必ずしもポータブルではない.	トークン, 鍵, スマートカード
バイオメトリクス (Who you are)	認証容易	効果. リプレイ攻撃. プライバシー侵犯. 特性の損傷／変化. 認識誤（正当なユーザも拒否, 不当なユーザを誤認証）	顔, 耳介, 指紋, 掌紋, 虹彩, 網膜, 音声, DNA 等
行為・行動 (What/How you do)	実現容易	精度が悪い 経時変化	音声認識, 筆跡

■ 表-6　本人認証の比較 ■

人情報は軍事機密等とは異なる側面を持ち，目的外利用，開示に関する権利の問題であるので，プライバシー保護は機密保護から独立させた．著作権等知的財産権保護も今後重要となる．

⑥保証（Assurance）：保証はいろいろな制度，仕組みの信頼の基礎となる．実装が適切に実施されていること，保護機構が適切であること，脅威に対する適切な耐性を備えることを保証し，実現レベル，実現手段を定義する．ポリシーで決まる．

特性間には**図-4**のように矢印の根元が先の方をサポートする関係がある．この他の重要な特性として非否認性（non-repudiation：自分の行為をそうではないと否定させないこと）がある．本稿ではこれは責任遡及性から派生するものとした．

(2)サポート機能：上記指標のベースとなる機能である．

①予防:暗号，認証（authentication），許諾（authorization），非否認，機密保護・著作権保護機能等

②検出・除去・軽減・回復：暗号，監査，侵入検知，隔離，冗長構成・分散化，総体性の保証，状態復帰等

いくつかに限定して説明する．

(3) TCB（Trusted Computing Base）：セキュリティ保護のベースとなる種々のハードウェア，ソフトウェア，システム構成情報，機構等をTCBと言う．

■ 暗号

暗号には**表-5**のようなものがある．米国Hunter College助教授Lester Hillの代数方程式暗号の発表（1929年）後，古典暗号とは異なる新しい暗号系が考えられるようになった．対称鍵暗号（秘密鍵暗号，共有鍵暗号ともいう）にはFeistel構造をベースとするDES（Data Encryption Standard：1974年NBSが採用），TDES（Triple DES），より安全なものとしてNISTに公募の結果採用されたAES（Advanced Encryption Standard）がある．公開鍵暗号は暗号化と復号化で鍵を別にし，暗号化（公開）鍵Kと暗号化関数E，復号化（秘密）鍵K'と復号化関数D，平文をM暗号文M'とすると，E（M;K）は計算容易であるが，D（M';K'）は計算が困難で，DE＝ED=恒等変換となる一方向関数を利用するものである．RSA，楕円曲線暗号，El Gamal等がある（RSA類似の概念は英国GHCQのCocksらの方が先であったが，軍事機密で発表されなかった）．公開鍵暗号は文書秘匿，改ざん検知，認証・署名，機密通信路の設定，秘密鍵配信等広く利用できる．また，暗号に類するものとして入力を一定の長さに変化する関数を用いるハッシュ（Hash）がある．これにはSHA-1, 256, 384, 512，MD-5等があり，IPsec，メッセージ認証等で重要な役割を果たす．

■ 認証と許諾[3)]

(1) 認証と許諾：authenticationとauthorizationがともに認証と訳されている場合があるが，authentication（認証）は識別子（ID：Identification）が真正か，登録されたものと一致しているかなどを照合検証するもので，authorization（許諾）はアクセスや利用の権利の認定・許可である．

(2) 本人認証：利用者が本人であることを確かめることである．**表-6**のような方法がある．パスワードは簡便に利用できるが，安易なパスワード付与はクラックされるので危険である．バイオメトリクス認証は顔，

耳介，指紋，掌紋，虹彩，網膜，音声，DNA等のパターン・特徴を利用する．ローカル利用には有効であるが，処理の複雑さ，身体条件の変化への対応，利用者の受容性，遠隔利用の場合の効率や盗聴・偽装等の問題がある．

(3) 情報へのアクセス：情報へのアクセスに関してはアクセス制御モデルと情報フロー制御モデルがある．アクセス制御はprincipal（または，subjectという：アクセスする主体．例：ユーザ）がobject（アクセスされる対象．例：ファイル）にアクセスし，利用するのを制御するもので，フロー制御は情報がprincipalに流れるのを制御するものである．アクセス権限はポリシーで決められる．

(4) アクセス・モードの定義：subjectとobjectの間のアクセス・モードはprincipal×objectのマトリックスにおいてアクセス・モードをエントリとするアクセス・マトリックス（**表-7**に例示）を利用すれば定義可能である．しかし，そのままではエントリーが多すぎ負荷が大になるので，アクセス制御リスト（ACL：Access Control List）またはケーパビリティ・リスト（capability list）が使われる．前者はアクセス・マトリックスの列方向のリスト，すなわち，すべてのobjectに対してアクセス権限のあるsubjectと，そのアクセス権限内容をリストとして持つ．利点は実装が容易なこと，オブジェクト指向に便利なことであるが，ユーザが離脱した場合の処理・廃棄が困難なこと，特定のsubjectを探すのに時間がかかるなどの問題がある．UNIXのファイル管理はこれに相当する．後者は行方向のリストで，権限の保持宣言が容易なこと，実行時チェックが容易なこと，代表の利用がしやすいことがある．欠点はあるobjectについてどのsubjectがアクセス権を持っているかを調べるのに時間がかかること，代表権限の消失処理を管理しにくいこと等である．

(5) アクセス・ポリシーのモデル：アクセスを形式的モデルにより記述する試みがなされている．たとえば，権限の支配関係による半順序集合を定義しアクセス許諾を定義する方法として，機密性に関するBell-La-Padulaモデル，完全性に関するBibaモデル，束モデル等があり，ルール・ベースで完全性をサポートするものとしてClark-Wilsonモデルがある．

	Obj-1	Obj-2	Obj-3	Obj-4
Subj-1	read	append	execute	read
Subj-2	write		read/execut	

■ 表-7　アクセス・マトリックス ■

■ システムの開発設計とセキュリティ

(1) システムのライフ・サイクルとセキュリティ：セキュリティはシステムのライフ・サイクルのあらゆるフェーズにかかわっており，システムのライフ・サイクルにその活動を組み込んでいく必要がある．たとえば，①開発設計モデルの選択，②プロジェクトあるいはチームのマネジメント，③各フェーズでのセキュリティへの対策，④各フェーズにおける仕様記述とフェーズ間での記述内容の整合，⑤コンポーネントとして組み込むソフトウェアのセキュリティの評価の問題等がある．どのフェーズであっても脅威あるいはリスクに対するアウェアネスがセキュリティ確立の第一歩である．

(2) 線形モデル：最も簡単なソフトウェアのライフ・サイクル・モデル（Royceによる）で，NIST，ISO/IEC，CMMI[5]等多くの標準で採用されている．その後，アプリケーション環境や基盤技術の速い変化に対応するため（Rapid）Prototyping，Spiral Model，Component Assembly Model，Concurrent Development，XP（Extreme programming），Clean Bench等のモデルや形式的方法が研究されている．ここでは，線形モデルをベースとして述べる．

(3)各フェーズでのセキュリティ[4]

①計画フェーズ：予備的な要求定義，リスクの予備的分析，セキュリティ・ポリシーとセキュリティ・アーキテクチャの設定をする．この段階は第1次的な分析と仕様の検討を行うもので，後にさらに詳細化する．問題は，(i) ユーザが必ずしも明確に自分自身の要求と重要度を把握していない／把握するのが困難なこと，(ii) ユーザと開発・設計（要求分析）者が言葉（国語という意味ではない）を共有していないこと，(iii) 自然言語で記述する場合，曖昧さがあること，(iv) 要求条件と実装の技術，期間，コストの関連が不明なことである．したがって，粗く漠然とした条件からワーキング・モデルを作成し，ユーザと対話しつつ，明確化する必要がある．契約のチェック，締結もこのフェーズで行う．

②設計：ライフ・サイクルの中でシステムの欠陥の固定に要するコストは後工程になるほど増大するので設計段階等前工程での対処が重要である．設計段階では，要求条件の詳細，要求仕様の策定，システムの機能／性能の詳細），プログラムの構造・構成，仕様する言語，導入ソフトウェア，プログラム・モ

ジュール間のインタフェース，他のエンティティとの通信方法・プロトコル，セキュリティ・ポリシーに従ったセキュリティ計画の詳細（リスク分析，セキュリティ機能，コンティンジェンシー計画等）決定等を行う．リスク分析にはFault-Mode Analysis，有限オートマトン，ペトリネット，リスク分析木，攻撃分析木等さまざまなモデルや手法がある．複数の異なる方式あるいは実装をするN-バージョン・システムによってリスク回避する方法もある．

③実装：設計を実際のプログラミング言語によるコードに変換する段階である．最近のプログラム開発では，設計と実装は明確に区別できない場合があるが，この段階では，たとえば，C，C++のバッファ・オーバフロー（GCCでは一部解決されている），浮動小数点演算の扱いのようにプログラム言語やその処理系に応じた処理，メモリ領域の初期化，アトミシティー，アルゴリズム適用方法，オープン・ソース等の導入コード，クローン・コード等のセキュリティに注意せねばならない．

④試験・統合：試験にはモジュール単体と（何段階かの部分的）統合試験がある．試験条件の試験計画は上流工程で検討しておく必要がある．100%の検査は一般に不可能である．セキュリティの保証レベルを定めておく必要がある．

⑤運用段階：性能の測定，攻撃等のモニタ，構成の管理，操作・運用者の管理，監査などを行う．

⑥廃棄：廃棄段階はあまり注目されないが，コピー機やディスクで近年話題となっているように，重要な問題である．廃棄時期と廃棄情報の選定，廃棄方法（捨てる，売る，寄付），契約の処理，情報の保存，記憶メディアの残存情報の処理等をせねばならない．

(4) 形式的方法：自然言語での記述には曖昧性があるので，形式的記述をして，次段階への自動変換，仕様・記述の正当性の検証，前段階との整合性の検証を行うことが考えられる．たとえば形式的記述ではZ，LOTOS，CSP等がある．ポリシー・レベルの形式的記述，机上での検証，機械的検証のレベルが考えられるが，100%形式化するのは困難で，巨大なシステムへの適用には複雑性の問題があり，研究の歴史は長いが現実には必ずしも成功していない．しかし，概念の整理，設計・実装の机上での照合検証，思考の節約と見通しの獲得に有効なことも多い．また，ISO/IEC15408でも軍用等クリティカルなアプリケーションに対する高度なレベルでは形式的な記述・検証が要求されている．今後形式的記述・検証技術は重要となろう．

■ DDoS攻撃と発信源追跡

犯罪跡の捜査，逃走経路を追跡し，犯人を特定し逮捕するのは実世界の犯罪への対処の常道である．サイバースペースがディペンダブルであるためには，同様なことが可能であることが望ましい．しかし，残念なことにインターネットでは記述のように発信アドレスはユーザが挿入でき，アドレス体系も追跡困難なものになっている．このような事態に対応するにはイングレス・フィルタリングや発信源探査などが考えられている．発信源探査にはパス構築を行う方法としてICMPパケットを使うiTrace，IPパケットをサンプリングしてパス情報を書き込むIPマーキング，パケット情報をハッシュし照合する方法（Hash Based Traceback）がある．これらの方法では集められたパス情報から攻撃パスを再構築する．パスを再構築しないで攻撃源側エッジルータを特定する方法もあり得る．ここでは省略するが，これらの方法には一長一短がある．ハンドラーやゾンビーという踏み台を使う間接的な攻撃では踏み台までで，真の攻撃者までは遡るのは困難である．インターネットがディペンダブルになるにはアドレスの体系，挿入，共通線信号，ネットワークの動的な管理制御の検討，そしてなによりもISPの協力が必要である．

標準化

ディペンダビリティに関する標準化は大きく3つに分けられる：①プロセスおよび管理に対する標準化，②技術仕様（プロトコル，構造，インタフェース）およびその他特性規準，③仕様記述に関する標準化である．現実の標準案，勧告，RFCをみるとかなり重複錯綜している．これは技術的に完全分離が困難なこと，各技術領域の検討にある種の完結性を持たせること，各検討グループが独立にテリトリーを決めること等によるものと思われる．標準には，またITU-T（国連の下部組織），ISO，IEC，EIA等のオーソリティによるde jure（bylaw）標準とIETFのRFCやビデオのVHS等のようにボランティアによるde facto（by fact：事実上の）標準がある．また，ANSI，JIS等の国家や地域標準，IEEE等学会，企業，コンソーシアムやフォーラムその他の組織標準がある．米国の軍の標準MIL，NIST標準およびカーネギーメロン大学のCMMI案等も大きな影響力を持っている．以下，いくつかについて簡単に紹介する：

(1) ソフトウェア・プロセスおよび管理：ISO 9000が代表的である．プロセスがよくなければ結果が悪いという考え方である．標準的な開発プロセスの設定や各段階での作業内容・標準，プロジェクト管理等を規定する．ISO 12207，1335，15504，17799，CMMI

（SSE-CMM等），NIST 800シリーズ等がある．
(2) 製品やサービスのセキュリティ要件：サービスや製品がどのような要件を満たすべきかを規定する．ISO 15408（Common Criteria），9126，ITU-T のX.509，ISO7498他，NIST800シリーズ，IEEE802.11i, IEEE982等
(3) 仕様記述：仕様の形式的記述に関するもので，data-flow-diagram，entity-relationship，UML，Z，LOTOSやプログラミング言語がこれに入る．

人材育成

米国においても我が国においても情報通信技術者，特にセキュリティや情報安全保障に関する知識を持った技術者の大幅な不足が懸念されており，その育成が焦眉の急である．米国でも指摘されているように，セキュリティの確保には従事技術者の計算機科学に関する知識，理論的な基盤，すなわち，大きなシステムや組織の設計や開発，情報環境の整備・運用にはより広く深い知識が必要である．しかし，多くの大学では実務教育が困難な状況にあり，一方，産業界では，一部の企業を除き基礎的な教育は困難である．このような状況の中で，工学院大学では1つの試みとして社会人と大学院生（他大学を含む）を対象とする教育プログラムが推進されている．ソフトウェア（アルゴリズム，セキュアOS，セキュアコーディング，XMLとデータベース，計算機アーキテクチャ），ネットワーク（構造・制御，プロトコル，QoS），ソフトウェア開発プロセス（ISO/IEC15504，CMM，PMBOK，スケジューリング，ヒューマン・エラー），セキュリティ技術の基本（概論，暗号，実例）の講義，IPsec，PKI，ファイアウォールその他実習を含む構成になっている．また，本講座の特色に，PBL（Problem Based Learning）がある．その1つは高エネルギー加速器研究推進機構（KEK）の共同研究施設のセキュリティ機能を提案することであり（同機構の協力を得ている），もう1つは情報処理推進機構（IPA），日本ネットワークセキュリティ協会（JNSA）の協力を得て進めているISO/IEC 15408仕様のセキュリティ文書作成である．さらに，日本e-ラーニング学会（JeLA）の協力を得てe-learning教材の検討を進めつつある．本計画は文部科学省科学技術振興調整費，新興分野人材養成プログラムの支援を受け，他大学および産業界の協力を得ながら進めている．なお，本講座の受講時間の一部をCISSP更新に必要な受講時間に組み入れることを個別申請した例がある．ITコーディネータの更新のマルチポイントにも組み入れられる．

まとめ

ディペンダビリティやセキュリティは情報通信に関する幅広い技術の上に成り立つ．本稿では紙数の都合からポリシーの決定，リスク／攻撃分析，暗号，IPsec，PKI，ファイアウォール，侵入検知，DoS攻撃，ウィルスやワーム，各種標準，法的問題等についてはほとんど触れることができなかった．本文でも述べたようにセキュア・システムの設計にはモデル化，形式的方法やセキュア・プログラミング，アルゴリズム等数理的な知識・素養が必要である．今後の計算機工学のカリキュラムにもディペンダビリティを取り込んでいく必要があろう．

謝辞　紙数の都合上個々の紹介は省略しますが，人材養成プロジェクトに協力をいただいている大学，IPA，KEK，JNSA，企業の方々に多大な感謝をいたします．また，ディペンダビリティやセキュリティに関連する問題について日ごろ議論し，本稿についても種々コメントをいただいた工学院大学倉光君郎助教授に感謝します．

参考文献
1) Laprie, J-C. : Dependability-Concepts, State-of-the-Art, Challenges, Critical Systems Conference 2001；http://www.csr.ncl.ac.uk/references/78/j.c._laprie.pdf
2) NIST SP-800シリーズ；http://www.nist.govから
3) Bishop. M. ed. : Computer Security, Addison Wesley(2003).
4) Graff M. G., van Wyk K.R. : Secure Coding, O'Reilly (2003).
5) Ahren D. M., Clouse, A., and Turner R. ed. : CMMI DIstilled, Addison Wesley(2003).
6) De Lemos, R., Weber, T. da S. and Camargo, J. B. Jr. ed.: Dependable Computing, Springer(2003).

（平成16年10月23日受付）