セキュアプログラミングの動向

伏見 諭

2010-08-31

目　　次

- セキュアプログラミング　ガイドライン／テキスト
- セキュアなWeb開発
- 書籍、テキスト等
- 国際規格
- 教育コース、資格など

このプレゼンでは、個別の技術課題の例を提供するわけではありません

セキュアプログラミング　ガイドライン

- “CWE/SANS Top 25 Most Dangerous Programming Errors”
  - SANS Top 20 attack vectors + MITRE's Common Weakness Enumeration
  - http://cwe.mitre.org/top25
- “CERT C Secure Coding Standards”（C++もあり）
  - https://www.securecoding.cert.org/confluence/display/seccode/CERT+C+Secure+Coding+Standard
  - http://www.jpcert.or.jp/sc-rules/（日本語訳）
  - SEIとCERT/CCは、これをもとに、「ソースコード解析ツールを活用したCERTセキュアコーディングスタンダードの有効性評価」という報告書も出している
- “The CERT Oracle Secure Coding Standard for Java”
  - https://www.securecoding.cert.org/confluence/display/java/The+CERT+Oracle+Secure+Coding+Standard+for+Java
  - Oracle/Sun自体のものとは別物
    - http://www.oracle.com/technetwork/java/seccodeguide-139067.html
- “Fundamental Practices for Secure Software Development”
  - SAFECode(Software Assurance Forum for Excellence in Code)
  - http://www.safecode.org/
  - “Software IntegrityControls – An Assurance-Based Approach to Minimizing Risks in the Software Supply Chain” (June, 14, 2010)も続編としてある

セキュアプログラミング　ガイドライン/調査/テキスト

- マイクロソフト社のSTRIDEモデル
  - http://msdn.microsoft.com/ja-jp/magazine/cc163519.aspx
- IATACのレポート
  - State-of-the-Art Report: “Software Security Assurance”
  - IATAC=Information Assurance Technology Analysis Center
  - セキュアプログラミング固有の話題は少ないが、セキュアな開発総体の鳥瞰図はしっかりしている
- セキュア・プログラミング講座 > C/C++言語編
  - http://www.ipa.go.jp/security/awareness/vendor/programmingv2/clanguage.html

（参考）役に立つか？

- NISTのVDB/SCAP
  - 脆弱性データベースであり、知識源とはなるが、直接的にセキュアプログラミングを提示するものではない
- ハッキング情報サイト
  - 上と同じ
- モバイル系、組込み系へ
  - ケースバイケース
- 一般的なコーディングガイド（コーディングルール）も、セキュリティ的に役に立つことが多い
  - 古い例としては：

    NASA SOFTWARE ENGINEERING LABORATORY SERIES SEL-94-003

    C STYLE GUIDE

    AUGUST 1994

（参考）私の古い資料集－１

- Secure Programming for Linux and UNIX HOWTO
- (最も知られているルール集）
  - http://www.dwheeler.com/secure-programs/
  - http:/www.linux.or.jp/JF/JFdocs/Secure-Programs-HOWTO/
- 「Secure Programming Cookbook for C and C++」の支援サイト
  - http://www.secureprogramming.com/
  - http://www.secureprogramming.com/?action=browse&feature=recipes

（参考）私の古い資料集－2

- Best Practices for Secure Development（Razvan Peteanuによる．やや古い(2001-Oct)が、しっかりした視点でかかれているので役にたつ）
  - (現在）http://www.linuxsecurity.com/resource_files/documentation/best_prac_for_sec_dev4.pdf
  - (旧）http://members.rogers.com/razvan.peteanu

（参考）私の古い資料集－3

- Secure Unix Programming Checklist (AusCERTのチェックリスト）
  - http://auscert.org.au/render.html?it=1975&cid=1920
- IPAの「セキュアプログラミング講座」解説とリスト（比較的充実している）
  - http://www.ipa.go.jp/security/awareness/vendor/programming/index.html
  - 「オープンソースのセキュリティ確保に関する調査報告書」も参考となる
    - http://www.ipa.go.jp/security/fy14/reports/oss_security/index.html

（参考）私の古い資料集－4

- OWASP（Open Web Application Security Project ）プロジェクト
  - http://www.owasp.org/index.jsp
  - http://jaist.dl.sourceforge.net/sourceforge/owasp/guide_v2a1.pdf
  - http://nchc.dl.sourceforge.net/sourceforge/owasp/OWASP_Top_Ten_2004_Japanese.pdf
- WAS （Web Application Security)フォーラム
  - http://www.wasf.net/
- Web securityについては次も参照（運用視点が強い）
  - http://www.w3c.org/Security/faq/www-security-faq.html
- Webセキュリティ解説については次の本も詳しい
  - 「技術者のためのプログラミングによるセキュリテイ強化ガイド」宮前竜也著、技術評論社、2002

（参考）私の古い資料集－5

- SANS Top 20 Vulnerabilities
  - http://www.sans.org/top20/
- 「セキュアプログラミング」の著者のサイト（ルールはプログラミングルールというより、開発ルール）
  - http://www.securecoding.org/
  - http://www.securecoding.org/companion/checklists/

（参考）私の古い資料集－6

- SQLに関するセキュリティチェックリスト
  - http://www.sqlsecurity.com/DesktopDefault.aspx?tabid=24
- SUNのセキュリティチェックリスト(Security Code Guidelines)
  - http://java.sun.com/security/seccodeguide.html

（参考）私の古い資料集－7

- 簡潔なルール（場合により、そのまま規約としてもよい）
  - NCSAの簡潔なセキュリティルール
    - http://archive.ncsa.uiuc.edu/General/Grid/ACES/security/programming/
  - CERT Top 10 Secure Coding Practices
    - http://www.csecurecoding.cert.org/confluence/display/seccode/tTop+10+Secure+Coding+Practices
  - ネット情報から（個人サイトのセキュリティプログラミングルール）
    - http://www.gadgety.net/shin/tips/unix/programming.html

（参考）DEP回避攻撃の資料

--ハードウェアDEPの問題点

ハードウェアDEPの基本的な考え方は、プロセッサがサポー
コード実行可能なメモリ領域とコード実行不可能なメモリ領
を利用して、意図しないコード実行を防止するというもので

メモリ破壊により発生する脆弱性は、一般的に、

(1)脆弱性を利用した制御フローの奪取

(2)スタック領域・ヒープ領域に読み込まれた攻撃コードの実

の2つのステップから成り立ちます。

ハードウェアDEPが有効化された環境下では、プログラムが
スタック領域・ヒープ領域はコード実行不可能となり、前述

Tips

→ Windows TIPS TOPへ
→ Windows TIPS全リストへ

XP SP2のデータ実行防止（DEP）機能を無効にする

→ 解説をスキップして操作方法を読む

デジタルアドバンテージ
2004/11/20

対象OS
Windows XP Service Pack 2

■Summary
■XP SP2には、スタックやヒープ上に置かれたプログラムの実行を防止するDEP機能がある。DEP機能を利用すると、バッファ・オーバーフローの脆弱性を突いてコードを実行するウィルスやワームを防止することができる。
■DEP機能は、Windows OSシステムに適用されるほか、ユーザー・プログラムに対して適用することもできる。
■古いバージョンのプログラムやデバイス・ドライバなどでは、DEP機能によって実行できなくなる場合がある。そのような場合はDEP機能を無効にすればよい。

http://www.atmarkit.co.jp/fwin2k/win2ktips/501dep/dep.html

--Return-into-libc攻撃によりハードウェアDEPをバイパス

しかしハードウェアDEPは、Return-into-libc攻撃と呼ばれる手法で簡単にバイパスできてしまう問題があり、攻撃方法が公表されています。

Return-into-libc攻撃とは、例えばスタックベースのバッファオーバーフローにおいて、親関数に戻る際に攻撃コード（shellcode）にジャンプするのではなく、何らかのAPIに直接ジャンプさせることにより、データ領域でのコード実行を行うことなく攻撃を成功させる手法です。

https://www.netsecurity.ne.jp/3_15315.html

Windowsのセキュリティ機能「DEP」を回避する新手法が公開される

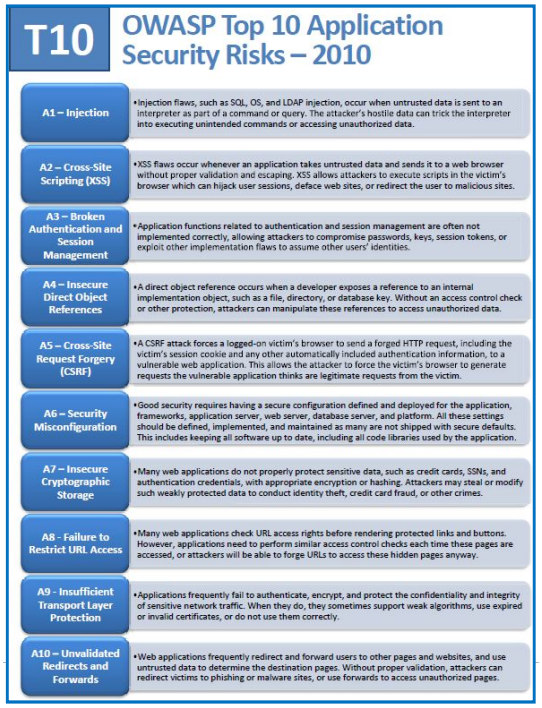
「攻撃の成功率が高まる」とセキュリティ研究者が予測（2010年03月04日）

Windowsのセキュリティ機能を回避する新手法が公開されたことで、同OSに対する攻撃が成功する可能性が高まった――セキュリティ・ベンダーの研究者が3月3日に指摘した。Googleのセキュリティ・ソフトウェア・エンジニア、ベレンドジャン・ウィーバー（Berend-Jan）氏は3月1日、WindowsのDEP（データ実行防止機能）が回避可能であることを示す概念実証コードを公開した。

http://www.computerworld.jp/topics/vs/175929.html

セキュアなWeb開発

- “OWASP TOP 10”
  - The Open Web Application Security Project
  - http://http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- PCI DSS（Payment Card Industry Data Security Standard）の基準
  - 要件6.5 すべての Web アプリケーション（内部、外部、アプリケーションへの Web 管理アクセス）を、「Open Web Application Security Project Guide」などの安全なコーディングガイドラインに基づいて開発する。
  - https://www.pcisecuritystandards.org/security_standards/pci_dss_download.htmlを参照



セキュアなWeb開発（続き）

- IPA（情報処理推進機構）の「セキュア・プログラミング講座・Webアプリケーション編」
  - http://www.ipa.go.jp/security/awareness/vendor/programmingv2/web.html
- IPA「安全なウェブサイトの作り方」日本語版（2010年1月20日改訂第4版公開）
  - http://www.ipa.go.jp/security/vuln/websecurity.html
  - 安全なウェブサイトの作り方(全92ページ、2.09MB)
  - セキュリティ実装 チェックリスト（Excel形式、33KB）
  - 英語版（2010年6月7日改訂第4版公開）How to Secure Your Web Site(98pages, 2.62MB)
  - 別冊：「安全なSQLの呼び出し方」日本語版（2010年3月18日公開）

（参考）ルールチェッカー

- Klocwork Insight（商用、C）、PMD（フリー、Java）など、ルールチェッカー／静解析ツールは多数あるが、今日のプレゼンでは省略
  - http://www.klocwork.com/jp/
  - http://pmd.sourceforge.net/
- ルールチェッカーの一般的な特質は、しばしば「厳しすぎる」ことである
- 他のチェッカー
  - 動解析ツールも併用されるが、セキュリティ固有目的では有効性が限られるか？
  - 脆弱性チェッカーは、別格として必要だが、新規コード対応というより、商用コードやOSS対応のものが多いか？　ファジー的なものは新規コードに対応。

マイクロソフトのSDLでは、“ライフサイクル全体を通した製品チームの活動は、ペネトレーション テストよりも脅威モデリング、コード レビュー、自動化ツールの使用、およびファジー テストに重点を置いています。後者の手段の方が、セキュリティのバグを防止または排除する上で、従来のアドホックなペネトレーション テストよりもはるかに厳密です。”としている

書籍

（挙げれば、たくさんありますが、とりあえず・・・）

- “C/C++セキュアコーディング”,R.C.Seacord
  - 日本語版はJPCERT/CC訳
- “Writing Secure Code”, M.Howard, D LeBlanc
  - 日本語版は、日経BPソフトプレス　上・下巻
- “Security on Rails”, B Powerski, D. Raphael
  - Pub. By Pragmatic Programmers LLC,(Ed. By C.Toporek)

(参考) Writing Secure Code第2版〈上〉プログラマのためのセキュリティ対策テクニック (マイクロソフト公式解説書)

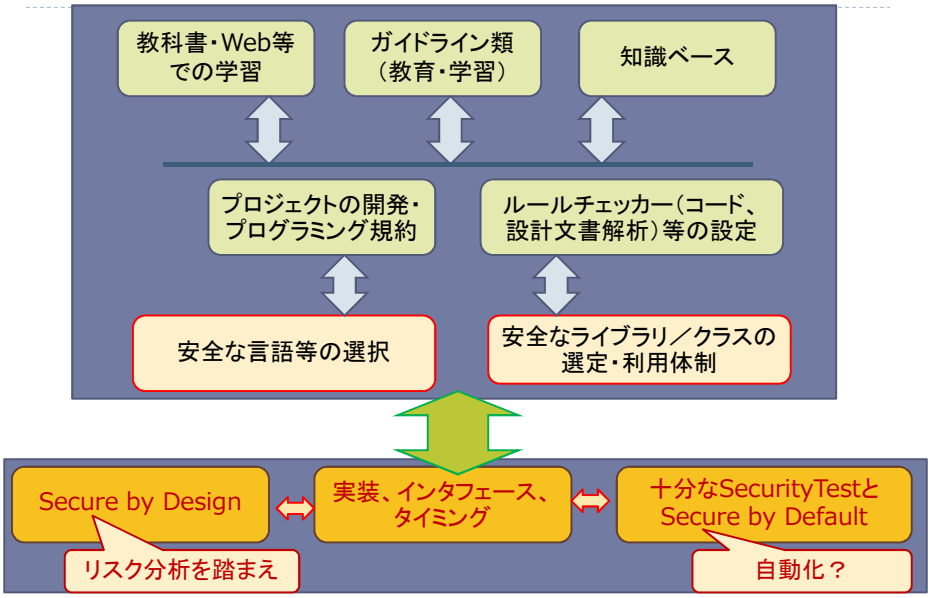
- 第1部 最先端のセキュリティ
  - セキュリティの必要性
  - セキュリティを積極的に取り込んだ開発プロセス
  - セキュリティの原則
  - 脅威のモデリング
- 第2部 セキュアなコーディングテクニック
  - バッファオーバーフロー
  - アクセス制御の考え方
  - 最小限の権限の原則
  - 暗号化技術の弱点
  - 機密データの保護
  - 検証なきデータはすべて有害と思え
  - 標準表記の問題
  - データベース入力の問題
  - Web固有の入力の問題
  - 国際化に伴う問題

並列プログラミングに伴うレースコンディション、デッドロックの問題は今後重要性を増すかWindowsではUAC、DEPへの対応をきちんと行う必要性

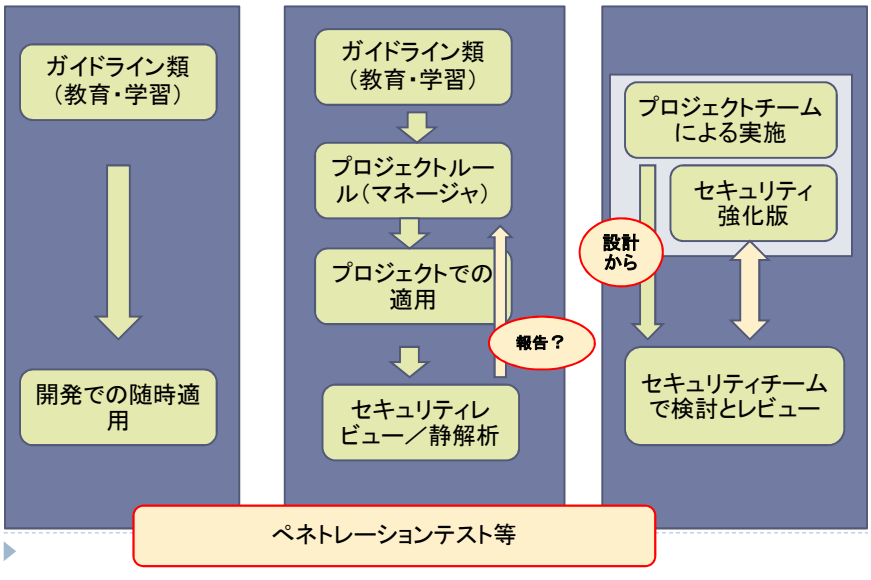
(参考) Writing Secure Code第2版〈下〉

- 第3部 さらにセキュアなコーディングテクニック
  - ソケットのセキュリティ
  - RPC、ActiveXコントロール、DCOMのセキュリティ
  - DoS攻撃への対処
  - .NETのセキュアなコーディング
- 第4部 その他のセキュリティトピック
  - プログラムのセキュリティテスト
  - セキュリティコードレビューの実施
  - インストール時のセキュリティ
  - プライバシーを組み込んだアプリケーションの開発
  - セキュリティのベストプラクティス
  - セキュリティ関連のドキュメントとエラーメッセージの作成
- 第5部 付録

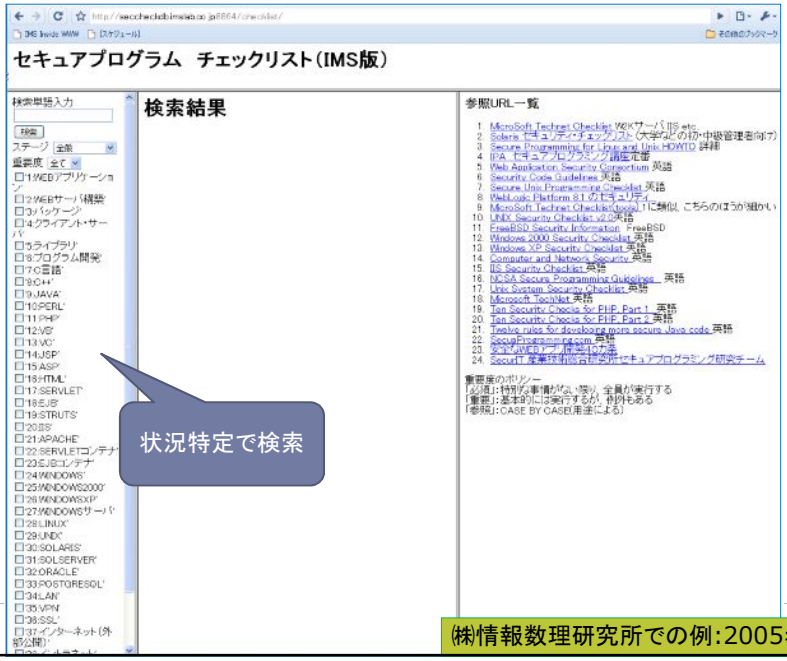
（参考）実施アプローチ（技術）



（参考）実施アプローチ（組織）



（参考）コーディングルールデータベース



㈱情報数理研究所での例:2005年頃

（参考）企業をまたぐ問題

- サプライチェイン問題
  - SAFECodeの “Software IntegrityControls – An Assurance-Based Approach to Minimizing Risks in the Software Supply Chain” (June, 14, 2010)は、これが主題
- OSS利用問題
  - OSSのセキュリティ問題への開発管理上の問題

国際規格

- 従来、ISMSやCCのいわば総論的な記述で止まっていたが、現在、より詳細化の方向での提案・ドラフトがいくつか進行中
  - 24772
  - 27034
  - 29193
  - 20004
  - 29147
  - (参考)24731
  - （ちなみに・・・・関連規格3件）

ISO/IEC TR 24772

- ISO/IEC DTR 24772 Information technology -- Programming languages -- Guidance to avoiding vulnerabilities in programming languages through language selection and use
  - 脆弱性の発生原因を列挙し、セキュリティガイドライン設定のためのガイドを提供している
  - SC22が制定

ISO/IEC 27034

- ISO/IEC FCD 27034-1 Information technology — Security techniques —Application security — Part 1: Overview and concepts
  - (引き続き、次の計画が出されている)
  - PART 2: Organization normative framework
  - PART 3: Application security management process
  - PART 4: Application security validation
  - PART 5: Protocols and application security control data structure
  - PART 6: Security guidance for specific applications
- ISMSの下でのアプリケーションセキュリティ開発の管理フレームワークをしめす(問題点が多い)

ISO/IEC 29193

- ISO/IEC WD 29193 Information technology — Security techniques —Secure system engineering principles and techniques
- 開発のプロセスごとに重要となるセキュリティエンジニアリングの観点を列挙している
  - また、ANNEX Aとして、重要なセキュリティエンジニアリング原則をカタログとして示している
  - 例: NEED-TO-KNOW原則
- 私見では、結構使えそう

ISO/IEC TR 20004

- ISO/IEC WD 20004 -- Information technology -- Security techniques –Secure software development and evaluation under ISO/IEC 15408 and ISO/IEC 18045
- CC と CEM の下での脆弱性分析の基本観点を示している
- USのCommon Weakness Enumeration (CWE) and the Common Attack Pattern Enumeration and Classification (CAPEC)との関連を考慮

ISO/IEC 29147

- ISO/IEC CD 29147 Information technology – Security techniques – Vulnerability Disclosure
- 安全な脆弱性情報の開示の方法を提示している

（参考）ISO/IEC TR 24731-1

- ISO/IEC TR 24731-1:2007 Information technology -- Programming languages, their environments and system software interfaces -- Extensions to the C library -- Part 1: Bounds-checking interfaces
  - 主として場ファーオーバーフロー対策のセキュアコーディング対応文字列関数
  - 実装例は多くなってきている（Visual C++ 2005以後など）
  - SC22が制定

ちなみに(その1)：　21827

- ISO/IEC 21827 **Information technology -- Security techniques -- Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®)**
  - ソフトウェア開発のCMMモデルをセキュリティエンジニアリング分野に適用したもの
  - リスク分析重視、運用（インシデント管理）と開発の結合管理などを特色とする

ちなみに(その2)： SQuaRE

- JTC1/SC7のWG6で開発中のソフトウェア品質モデル
  - 大きな規格体系
    - ISO/IEC 25000series
  - 元の（著名な）ISO/IEC 9126規格等の大幅改訂版
    - ソフトウェアコミュニティの品質概念は基本的にはこの規格に依拠している
  - セキュリティ関連の特性も品質の重要部分を占める

ちなみに(その3)： Software Assurance

- JTC1/SC7のWG7で開発中の規格
  - ISO/IEC 15026 Systems and software engineering — Systems and software assurance:
    - Part 1: Concepts and vocabulary
    - Part 2: Assurance case
    - Part 3: System integrity levels
    - Part 4: Assurance in the life cycle
  - 元の「インテグリティレベル規格」を拡張・改訂したもの
  - ソフトウェアの（品質等の）保証のフレームワークを規定する

教育コース、資格など

- (ISC)2でのISSLP資格制度の開始
  - 次ページ以下に詳述
- GIAC
  - GIAC Secure Software Programmer - .NET (GSSP-NET)
  - GIAC Secure Software Programmer - Java (GSSP-JAVA)
- CompTIA Security+
  - プログラミングに特化したものは無いようだ
- 日本の情報処理技術者試験
  - 情報セキュリティスペシャリスト試験
    - プログラミング中心ではない
- 一般向け教育コース
  - コンピュータメーカ各社
  - JISA ICTカレッジ(「セキュリティの構築」など、基礎的コース)
  - シマンテック ーー> SCSP等の制度は廃止(2007)
  - その他？

(参考) (ISC)^{2}のイベント

- **SecureSDLC: Building Security into the Software Lifecycle (Nov 04 @Washington, DC)**
- 9:00am - 10:00am: Avoiding the Most Dangerous Software Security Weaknesses – the 2010 Top 25" - Robert Martin-Mitre
- The talk will focus on the 2010 update to the SANS/CWE list of the Top 25 Most Dangerous Software Errors, which is the "minimal due-care standard" for developing secure applications in many large enterprises, and is used by the State of New York and DTCC to mandate that application security be addressed in procurement contracts.
- 10:00am - 10:15am: Break/Visit Sponsors
- 10:15am - 11:00am: Tripwire Session
- 11:00am - 12:00pm: Federal/Government Focused Panel
- 12:15pm - 1:30pm: Lunch
- 1:30pm – 2:30pm: Into the Rabbit Hole: Execution Flow-Based Web Application Testing -Rafal Los, HP
- Since the caveman first fashioned a spear, humans have been using tools to make them more efficient and effective. Unfortunately, today's analysts often misunderstand the role that tools play in testing web applications. While tools can be quite good at mapping a web application's attack surface, there is still much human analysis that must be done to find the elusive defects that lie just below the surface. That human analysis has been daunting and irregular ... until now.
- The answer is an execution-flow-based approach to application security testing. By first understanding the application logic and execution flow, it is possible to completely map a web application's attack surface, and therefore fully test the application. Along the way, we will cover the principles of data-flow analysis, application process mapping and building execution-flow diagrams (EFDs), which together form a complete picture of the web application
- 2:30pm - 3:00pm: Break/Visit Sponsors
- 3:00pm - 4:00pm: Deloitte Session
- 4:00pm – 5:00pm: Tom Volpe

Home ▸ Certification Programs ▸ CSSLP

Certified Secure Software Lifecycle Professional

CSSLP

The Certified Secure Software Lifecycle Professional (CSSLP) is the only certification in the industry that ensures security is considered throughout the entire lifecycle.

It's no secret that security is not being addressed from a holistic perspective throughout the software lifecycle. Some **80%** of all security breaches are application related equating to more than 226 million records being disclosed and fines reaching astronomical amounts. Together we are building security into the lifecycle, one CSSLP at a time.

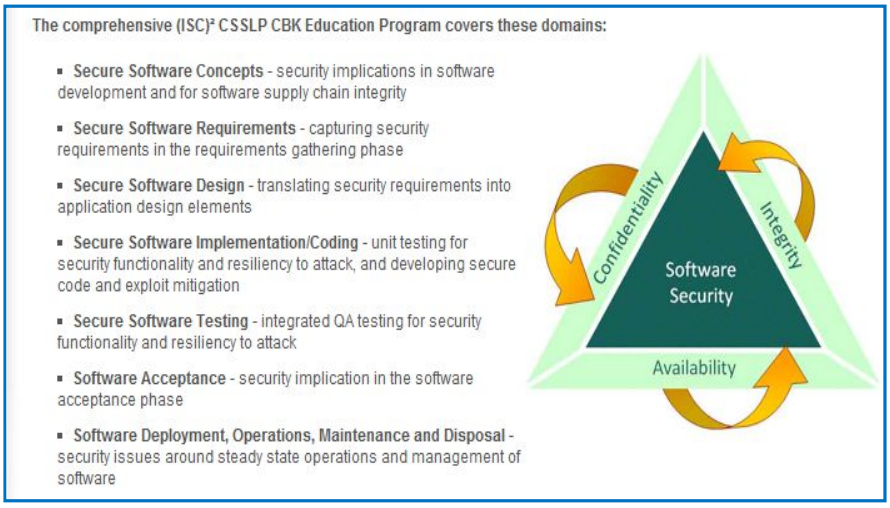
COMPUTER BASED TESTING NOW AVAILABLE FOR THE CSSLP

- CSSLP is for everyone involved in the Software Lifecycle with at least **4 years experience**.
- Learn more about **who should become a CSSLP?**
- **New! Download a brochure** now to learn more about the CSSLP.

The following domains make up the CSSLP CBK focus on the need for building security into the SDLC:

- **Secure Software Concepts** - security implications in software development and for software supply chain integrity
- **Secure Software Requirements** - capturing security requirements in the requirements gathering phase
- **Secure Software Design** - translating security requirements into application design elements
- **Secure Software Implementation/Coding** - unit testing for security functionality and resiliency to attack, and developing secure code and exploit mitigation
- **Secure Software Testing** - integrated QA testing for security functionality and resiliency to attack
- **Software Acceptance** - security implication in the software acceptance phase
- **Software Deployment, Operations, Maintenance and Disposal** - security issues around steady state operations and management of software

https://www.isc2.org/csslp/default.aspx



The comprehensive (ISC)² CSSLP CBK Education Program covers these domains:

- **Secure Software Concepts** - security implications in software development and for software supply chain integrity
- **Secure Software Requirements** - capturing security requirements in the requirements gathering phase
- **Secure Software Design** - translating security requirements into application design elements
- **Secure Software Implementation/Coding** - unit testing for security functionality and resiliency to attack, and developing secure code and exploit mitigation
- **Secure Software Testing** - integrated QA testing for security functionality and resiliency to attack
- **Software Acceptance** - security implication in the software acceptance phase
- **Software Deployment, Operations, Maintenance and Disposal** - security issues around steady state operations and management of software

CBK for CSSLP