# Systematyczna analiza bezpieczeństwa przy projektowaniu aplikacji

Wykład II

## Analiza bezpieczeństwa

- identyfikacja stron procesu,
- identyfikacja lokacji w procesie,
- identyfikacja akcji w procesie,
- identyfikacja urządzeń w procesie,
- identyfikacja i określenie wartości cennych zasobów,
- określenie zagrożeń i analiza zachowania komponentów oprogramowania,
- oszacowanie ryzyka w terminach stopnia zagrożenia i prawdopodobieństwa zajścia,
- określenie na jak długo analiza jest aktualna.

## Strony procesu

1. normalni uczestnicy interakcji,
2. zaufane trzecie strony,
3. dostarczyciele oprogramowania,
4. dostarczyciele systemu,
5. konserwujący oprogramowanie,
6. konserwujący system,
7. osoby opiekujące się budynkami, ulicami itp.,
8. potencjalni intruzi (atakujący).

# Lokacje, akcje, urządzenia...

- Lokacje: budynki, pomieszczenia, trasy itp.
- Akcje: proces działania oprogramowania (projekt)
- Urządzenia: komputery, karty, telefony, nadajniki, odbiorniki, łącza, itp.

# Model atakującego

- skąd pochodzi atakujący
  - któryś uczestnik,
  - zaufana trzecia strona,
  - twórca oprogramowania,
  - dostawca oprogramowania,
  - ktoś spoza wyróżnionych osób,
  - ...,

# Model atakującego

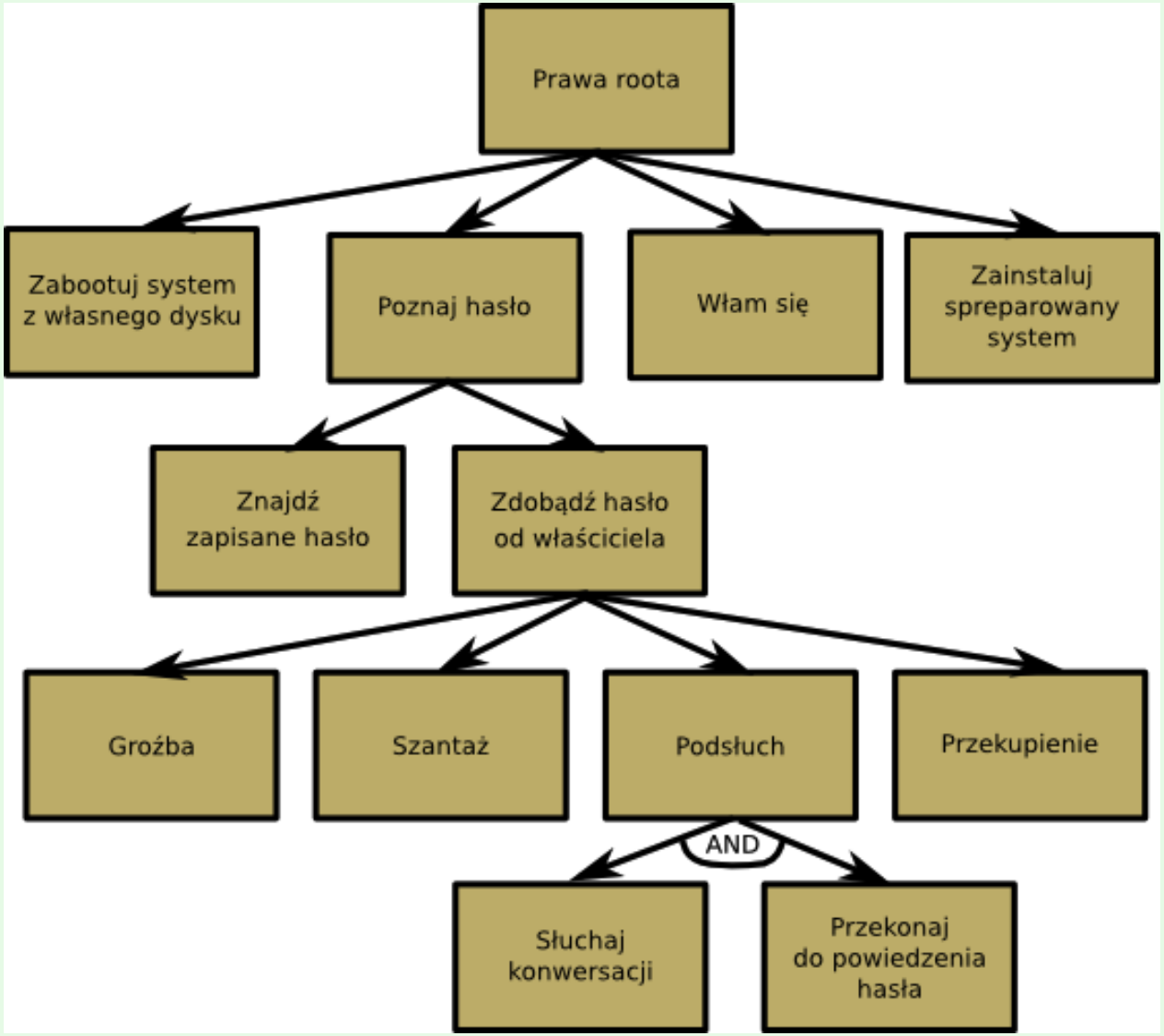
- jakie ma możliwości
  - możliwe miejsce ataku,
  - możliwy sposób ataku (aktywny, pasywny atak, atak statyczny, itp.),
  - zasoby atakującego (moc obliczeniowa, pamięć, itp.)

# Cenne zasoby

- zwykle wynikają z analizy CIA lub innych wymagań bezpieczeństwa,
- typowe cenne zasoby:
  - prywatne dane,
  - anonimowość osoby,
  - własność osoby/firmy,
  - dostępność usługi,
  - autentyczność informacji,
  - ...

# Określenie zagrożeń – drzewa ataków

- korzeń – docelowy cenny zasób,
- poddrzewa – sposoby na uzyskanie danego zasobu
- dwa rodzaje wierzchołków:
  - AND – atak możliwy, gdy wszyscy synowie zrealizowani,
  - OR – atak możliwy, gdy któryś z synów zrealizowany.


Prawa roota
Zabootuj system z własnego dysku
Poznaj hasło
Włam się
Zainstaluj spreparowany system
Znajdź zapisane hasło
Zdobądź hasło od właściciela
Groźba
Szantaż
Podsłuch
Przekupienie
AND
Słuchaj konwersacji
Przekonaj do powiedzenia hasła

## Szacowanie ryzyka

- Można w drzewach ataków zaznaczać, czy możliwy/niemożliwy atak
- Można bardziej skomplikowane rzeczy:
  - łatwy – trudny,
  - drogi – tani,
  - legalny – nielegalny,
  - ze specjalnym sprzętem – bez specjalnego sprzętu,
  - ...
- model atakującego pozwala na znalezienie kryteriów j.w.

# Lista zagrożeń wg *19 deadly...*

- Przekroczenie buforu (Buffer overruns)
- Napisy formatujące (Format strings)
- Przekroczenie zakresu liczb (Integer overflows)
- Wstawienie SQL-a (SQL injections)
- Wstawienie polecenia (Command injections)
- Brak obsługi błędów (Failing to handle errors)

## Lista zagrożeń wg *19 deadly...*

- Skrypty wieloserwerowe (Cross-site sripting)
- Brak zabezpieczenia ruchu sieciowego (Failing to protect network traffic)
- Używanie magicznych URL-i i niewidocznych pól w formularzach (Use of magic URLs and hidden form fields)
- Nieodpowiednie użycie SSL i TLS (Improper use of SSL and TLS)
- Używanie słabych systemów opartych na haśle (Use of weak password-based systems)
- Brak bezpiecznego zapisu i zabezpieczenia danych (Failing to store and protect data securely)

## Lista zagrożeń wg *19 deadly...*

- Wyciek informacji (Information leakage)
- Nieodpowiedni dostęp do plików (Improper file access)
- Ufanie DNS-owi (Trusting DNS)
- Wyścigi (Race conditions)
- Nieautentyfikowana wymiana kluczy (Unauthenticated key exchange)
- Silne generatory liczb losowych (Cryptographically strong random numbers)
- Kiepska funkcjonalność (Poor usability)

# Przekroczenie buforu

- używaj bezpiecznych funkcji dostępu do bufora,
- używaj ochrony zapewnianej przez kompilator (np. opcja /GS kompilatora, StackGuard, analiza statyczna),
- używaj ochrony na poziomie systemu operacyjnego (np. DEP, PaX),
- nie myśl, że to wystarczy,

# Przekroczenie buforu c.d.

- nie twórz nowego kodu korzystającego z niebezpiecznych funkcji,
- rozważ uaktualnienie kompilatora,
- rozważ usunięcie niebezpiecznych funkcji ze starego kodu,
- rozważ używanie opakowań (np. z C++) zamiast bezpośredniego manipulowania na buforach.

# Napisy formatujące

- używaj stałych napisów formatujących lub napisów z zaufanych źródeł,
- nie sprawdzaj i nie ograniczaj dostępu locale do poprawnych wartości,
- nie używaj bezpośrednio tekstu wprowadzonego przez użytkownika jako napisu formatującego,
- rozważ użycie języków wyższego poziomu, które są zwykle mniej podatne na to zagrożenie.

# Przekroczenie zakresu liczb

- sprawdzaj wszystkie obliczenia związane z alokacją pamięci,
- sprawdzaj wszystkie obliczenia związane z indeksami w tablicach,
- używaj liczb bez znaków do obliczania indeksów w tablicach i obliczania rozmiarów alokowanych buforów,
- nie przyjmuj, że języki inne niż C/C++ są odporne na ten rodzaj błędu.

# Wstawienie SQL-a

- poznaj dokładnie DBMS, którego używasz:
  - czy masz do dyspozycji procedury składowe?
  - jak wyglądają komentarze?
  - czy atakujący może wywoływać rozszerzone operacje?
- sprawdzaj poprawność danych wejściowych,
- bądź świadom wiarygodności danych wejściowych,

# Wstawienie SQL-a c.d.

- używaj parametryzowanych zapytań (prepared statements),
- przechowuj informacje o połączeniu z bazą poza aplikacją,
- nie polegaj wyłącznie na filtrowaniu „złych słów”,
- nie dowierzaj danym wejściowym służącym do tworzenia zapytań SQL,

# Wstawienie SQL-a c.d.

- nie używaj łączenia napisów do budowania zapytań (nawet przy wywoływaniu procedur składowych),
- nie wywołuj procedur składowych z parametrami, którym nie ufasz,
- nie wykonuj prostego podwajania znaków cudzysłowu,
- nie łącz się z bazą danych jako root,

# Wstawienie SQL-a c.d.

- nie wstawiaj hasła do aplikacji lub napisu tworzącego połączenie,
- nie zapisuj informacji konfiguracyjnych bazy w głównym katalogu serwisu WWW,
- rozważ zabronienie dostępu bezpośredniego do tabel użytkownika i sięgaj do nich tylko przez procedury, wywołuj je jako parametryzowane zapytania.

## Wstawienie polecenia

- wykonuj sprawdzanie wszystkich danych wejściowych przed przekazaniem ich do procesora poleceń,
- obsługuj bezpiecznie niepowodzenie przy sprawdzaniu danych wejściowych,
- nie przekazuj do żadnego interpretatora poleceń niesprawdzonych danych (nawet jeśli myślisz, że przekazujesz dane),
- nie stosuj list wykluczających (chyba, że jesteś 100% pewien, że pokrywają one 100% sytuacji),
- rozważ rezygnację ze sprawdzania danych wejściowych przez filtrowanie regularnymi wyrażeniami, lepiej napisz prosty i klarowny walidator ręcznie.

# Brak obsługi błędów

- sprawdzaj wynik każdej funkcji związanej z bezpieczeństwem,
- sprawdzaj wynik każdej funkcji zmieniającej ustawienia użytkownika lub ustawienia całego systemu,
- rób wszystko, aby łagodnie wychodzić z błędnych sytuacji – pomaga to unikać ataków DoS,
- nie wprowadzaj generycznego łapania wszystkich wyjątków (chyba że masz dobry powód po temu), to może maskować poważne błędy,
- informacja o błędach nie powinna trafiać do niezaufanych użytkowników.

# Skrypty wieloserwerowe

- sprawdzaj wejście ze strony WWW, czy jest poprawne i czy można mu ufać,
- stosuj kodowanie HTML wszystkich danych pochodzących od użytkownika,
- nie wypisuj danych pochodzących z WWW bez uprzedniego sprawdzenia ich poprawności,
- nie przechowuj w ciasteczkach ważnych informacji,
- rozważ wprowadzenie tak wielu osłon, jak tylko to możliwe,

# Brak zabezpieczenia ruchu sieciowego

- wykonuj ciągłą autentyfikację całego ruchu produkowanego przez twoją aplikację,
- używaj silnego mechanizmu przy początkowej autentyfikacji,
- szyfruj wszystkie dane, dla których ważne jest zachowanie poufności, myl się na korzyść poufności,
- korzystaj z SSL/TLS, gdzie się da, do swoich potrzeb kryptograficznych,

# Brak zabezpieczenia ruchu sieciowego c.d.

- nie wkodowuj na stałe żadnych kluczy,
- nie myśl, że operacja XOR o stałym kluczu, cokolwiek szyfruje,
- nie pozwól, aby efektywność zwyciężyła nad potrzebą szyfrowania, szyfrowanie w locie jest tanie,
- zastanów się nad użyciem technik na poziomie sieci (t.j. ścian przeciwogniowych, wirtualnych sieci prywatnych, wyrównywaczek obciążenia itp.)

## Używanie magicznych URL-i i niewidocznych pól w formularzach

- przy testowaniu aplikacji testuj całe wejście pochodzące z WWW (w tym formularze) z użyciem złośliwych danych,
- jeśli nie korzystasz z przetestowanych metod kryptograficznych, to postaraj się dobrze rozumieć siły i słabości swojego podejścia,
- nie umieszczaj tajnych danych w żadnej konstrukcji HTTP czy HTML-a, nie upewniwszy się wcześniej, że są one opakowane w jakąś formę kryptograficznego zabezpieczenia (IPsec, SSL lub na poziomie aplikacji),

## Używanie magicznych URL-i i niewidocznych pól w formularzach

- nie ufaj żadnym danym, poufnym lub jawnym, pochodzącym z formularza, gdyż złośliwi użytkownicy mogą je zmienić niezależnie od zastosowania SSL,
- nie myśl, że ponieważ użyłeś kryptografii, to twoja aplikacja jest bezpieczna – atakujący znajdzie sposób, aby ją obejść.

# Nieodpowiednie użycie SSL i TLS

- używaj ostatniej dostępnej wersji SSL/TLS, w kolejności preferencji: TLS 1.1, TLS 1.0, SSL 3,
- korzystaj, w miarę możliwości, z listy dopuszczonych certyfikatów,
- upewnij się, że przed wysłaniem danych certyfikat odbiorcy jest prześledzony aż do zaufanego CA i że obecnie trwa okres jego ważności,
- sprawdź, że spodziewana nazwa maszyny znajduje się w odpowiednim miejscu certyfikatu odbiorcy,

# Nieodpowiednie użycie SSL i TLS c.d.

- nie korzystaj z SSL2 – poważne luki w kryptografii,
- ostrożnie korzystaj z biblioteki SSL/TLS,
- nie ograniczaj się do sprawdzenia wyłącznie nazwy w certyfikacie,
- rozważ korzystanie z respondera OCSP do sprawdzania, czy certyfikat nie był unieważniony,
- rozważ załadowanie CRL po utracie ważności przez aktualnie używaną listę.

# Używanie słabych systemów opartych na haśle

odtąd omawiaća - zrobic numeracje stron

- zapewnij, aby nie można było łatwo podejrzeć hasła przy autentyfikacji,
- zawsze informuj o nieudanej próbie zalogowania w ten sam sposób, niezależnie od tego, jaki był faktyczny powód,
- rejestruj nieudane próby zalogowania,
- do przechowywania informacji o haśle używaj wypróbowanej, jednokierunkowej funkcji haszującej z dodatkowym zakłóceniem,

## Używanie słabych systemów opartych na haśle c.d.

- udostępnij posiadaczom haseł bezpieczny mechanizm jego zmiany,
- nie stwarzaj ludziom z BOK-u możliwości łatwego skasowania hasła,
- nie dostarczaj oprogramowania z założonym domyślnym kontem i domyślnym hasłem; wprowadź specjalną procedurę ustawiania hasła domyślnego konta przy pierwszym uruchomieniu systemu,

# Używanie słabych systemów opartych na haśle c.d.

- nie przechowuj haseł otwartym tekstem,
- nie przechowuj haseł w kodzie programu,
- nie rejestruj haseł z nieudanych logowań,
- nie dopuszczaj krótkich haseł,

# Używanie słabych systemów opartych na haśle c.d.

- zastanów się nad zastosowaniem kosztownego obliczeniowo algorytmu haszowania (np. PBKDF2),
- zastanów się nad zastosowaniem wielorakiej autentyfikacji,
- zastanów się nad zastosowaniem protokołów autentyfikacji typu "zero knowledge",
- zastanów się nad zastosowaniem haseł jednorazowych przy dostępie z systemów o niepewnej wiarygodności,

## Używanie słabych systemów opartych na haśle c.d.

- zastanów się nad zastosowaniem programowego sprawdzania siły haseł,
- zastanów się nad wprowadzeniem instrukcji, jak konstruować silne hasła,
- zastanów się nad wprowadzeniem mechanizmu automatycznego resetowania hasła po poprawnej odpowiedzi na zadane wybrane pytanie.

# Brak bezpiecznego zapisu i zabezpieczenia danych c.d.

- zastanów się nad prawami dostępu nakładanymi przez twój program i prawami dostępu dziedziczonymi z systemu operacyjnego,
- pamiętaj, że pewne dane nie powinny nigdy pojawiać się na serwerze produkcyjnym ogólnego przeznaczenia (np. długoterminowe klucze prywatne X.509), lepsze miejsce to specjalnie opracowane urządzenie sprzętowe używane tylko do podpisywania,

# Brak bezpiecznego zapisu i zabezpieczenia danych c.d.

- użyj wszelkich możliwości systemu operacyjnego, aby zabezpieczyć poufne i ważne dane,
- używaj odpowiednich praw dostępu (np. wbudowanych w system lub korzystających z ACL), jeśli musisz przechowywać ważne dane,
- po skorzystaniu z tajnych informacji usuń je z pamięci,
- wyzeruj pamięć przed zwolnieniem,

# Brak bezpiecznego zapisu i zabezpieczenia danych c.d.

- nie twórz plików z globalnym prawem do zapisu w Linuksie, Mac OS X czy Uniksie,
- nie twórz obiektów z wpisanym w ACE pełną kontrolą dla każdego lub zapisem dla każdego,
- nie przechowuj żadnych kluczy w strefie zdemilitaryzowanej (DMZ),

# Brak bezpiecznego zapisu i zabezpieczenia danych c.d.

- nie umieszczaj żadnych tajnych danych w kodzie aplikacji (haseł, kluczy, łańcuchów znaków do łączenia z bazą danych itp.),
- nie umieszczaj żadnych tajnych danych w kodzie przykładowych aplikacji t.j. w dokumentacji lub książce kucharskiej programowania,
- nie twórz własnych "tajnych" algorytmów szyfrowania,

## Brak bezpiecznego zapisu i zabezpieczenia danych c.d.

- zastanów się nad zaszyfrowaniem danych, które nie mogą być odpowiednio zabezpieczone przez ACL, oraz podpisaniem danych, aby zabezpieczyć je przed modyfikacją,
- zastanów się, czy nie możesz po prostu nie zapisywać sekretów (może da się pozyskać sekret od użytkownika w czasie rzeczywistym?)

# Wyciek informacji

- określ, kto powinien mieć dostęp do jakich informacji o błędach i stanie aplikacji,
- korzystaj z możliwości udostępnianych przez system operacyjny takich jak prawa dostępu i ACL,
- stosuj kryptografię do zabezpieczania ważnych danych,
- nie udostępniaj informacji o stanie systemu niezaufanym użytkownikom,

# Wyciek informacji

- nie podawaj bardzo dokładnych informacji czasowych przesyłając szyfrowane dane (jeśli trzeba, zmniejsz precyzję i/lub podaj te dane w zaszyfrowanej części),
- rozważ zastosowanie mniej powszechnych zabezpieczeń w systemie operacyjnym, np. szyfrowania plików,
- rozważ zastosowanie oprogramowania kryptograficznego zaprojektowanego z myślą o przeciwdziałaniu atakom czasowym,
- rozważ zastosowanie modelu Bella-LaPaduli (najlepiej w jakimś istniejącym mechanizmie).

# Nieodpowiedni dostęp do plików

- ściśle pilnuj dozwolonych nazw plików,
- nie przyjmuj na ślepo nazw plików (zwłaszcza na serwerach),
- rozważ zapisywanie plików tymczasowych w tymczasowym katalogu użytkownika.

# Ufanie DNS-owi

- stosuj kryptografię do identyfikacji serwerów i klientów (np. używając SSL),
- nie ufaj informacjom z DNS-u,
- zastanów się nad wprowadzeniem IPseca dla systemów, na których będzie działała twoja aplikacja.

## Wyścigi

- pisz kod, który nie zależy od efektów ubocznych,
- uważnie pisz procedury obsługi sygnałów,
- nie modyfikuj globalnych zasobów bez ich blokowania,
- zastanów się nad tworzeniem plików tymczasowych w obszarze pamięci przypisanej użytkownikowi, a nie w przestrzeni globalnie zapisywalnej.

# Nieautentyfikowana wymiana kluczy

- pamiętaj, że sama wymiana kluczy zwykle nie jest bezpieczna – należy także autentyfikować drugą stronę,
- przy ustanawianiu sesji korzystaj z rozwiązań z półki np. SSL/TLS,
- upewnij się, że przeczytałeś wszystkie rzeczy napisane drobnym drukiem, aby zapewnić, że wszystkie strony są odpowiednio mocno zautentyfikowane,
- jeśli upierasz się przy zastosowaniu rozwiązań domowej roboty, zawołaj kryptografa.

# Silne generatory liczb losowych

- jeśli możliwe, używaj systemowego kryptograficznego generatora liczb losowych (CSPRNG),
- upewnij się, że inne generatory kryptograficzne mają nasiono o co najmniej 64 bitach entropii albo lepiej 128,
- nie stosuj niekryptograficznych generatorów liczb losowych,
- w sytuacjach, gdy potrzebna jest duża pewność, używaj sprzętowych generatorów liczb losowych.

# Kiepska funkcjonalność

- staraj się rozumieć potrzeby bezpieczeństwa swoich użytkowników, podawaj im odpowiednie informacje pomocne w wykonywaniu ich zadań,
- tam gdzie to możliwe, ustaw domyślną konfigurację tak, aby była ona bezpieczna,
- podawaj proste i łatwe do zrozumienia komunikaty oraz dopuszczaj stopniowe poluźnienie dyscypliny tajności przy udziale bardziej zaufanych użytkowników lub administratorów,
- czytelnie zachęcaj użytkowników do wykonywania operacji związanych z bezpieczeństwem,

# Kiepska funkcjonalność

- nie wypisuj komunikatów z użyciem fachowej terminologii,
- nie wypisuj zbyt długich komunikatów,
- nie ułatwiaj użytkownikom wyrządzania sobie szkody (niebezpieczne opcje należy ukrywać),
- zapewnij metody selektywnego odbezpieczania aplikacji, ale zawsze jawnie i jasno opisuj konsekwencje.

# Proponowana lektura

- Gyrd Brændeland and Ketil Stølen, *Using model-based security analysis in component-oriented system development*, QoP '06: Proceedings of the 2nd ACM workshop on Quality of protection, 2006, http://doi.acm.org/10.1145/1179494.1179498, ACM Press
- Bruce Schneier, *Attack Trees* http://www.schneier.com/paper-attacktrees-ddj-ft.html
- Michael Howard, David LeBlanc, John Viega, *The 19 Deadly Sins of Software Security*, McGraw-Hill, 2005