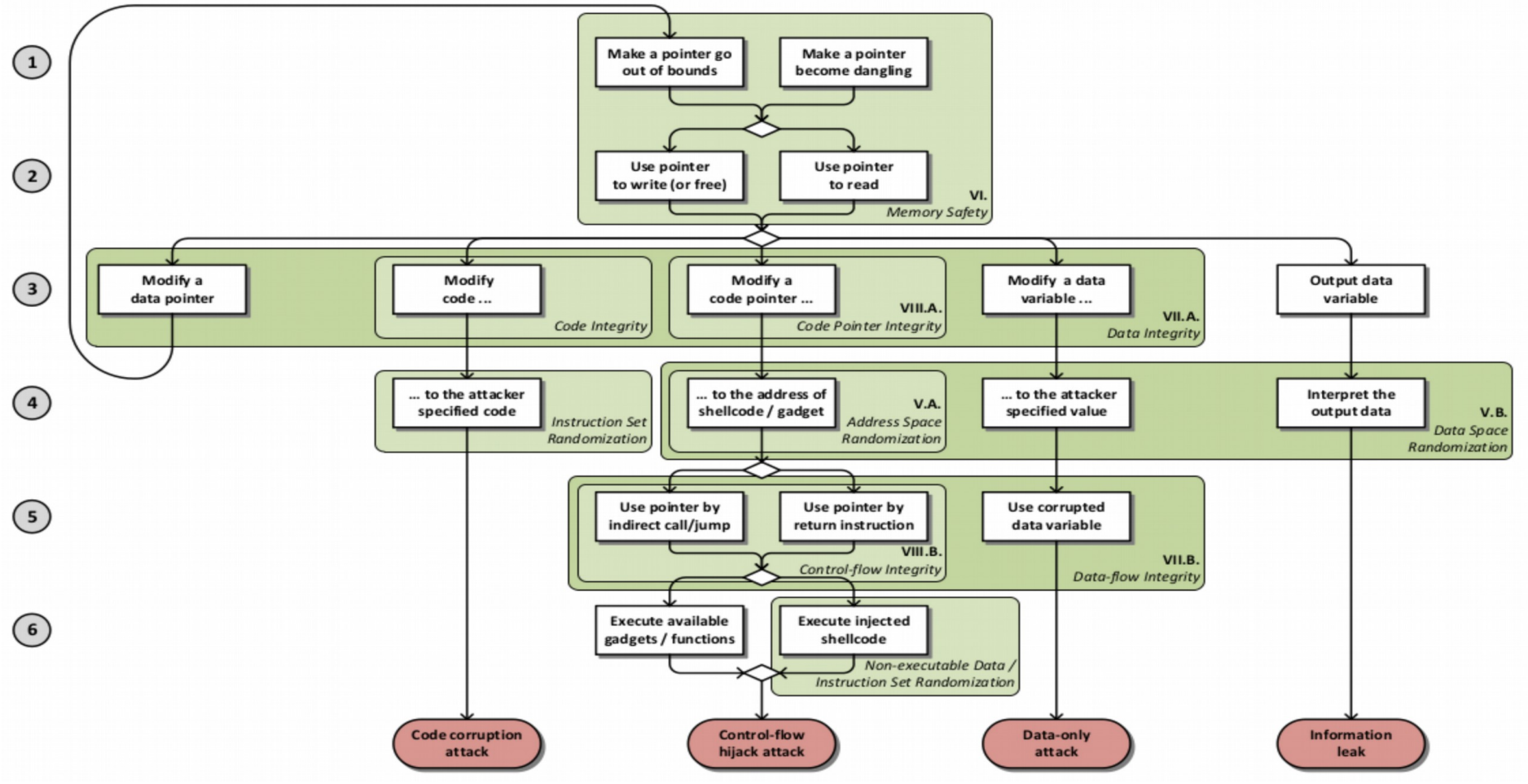
AUTOSAR Adaptive – zagadnienia bezpieczeństwa

Możliwe typy ataków na oprogramowanie

- Jedna z możliwych kwalifikacji rodzajów ataków, obejmuje:
  - Ataki na przełamanie kodu (ang. code corruption attack)
  - Przejęcie kontroli wykonania (ang. control-flow hijack attack)
  - Atak na dane (ang. data-only attack)
  - Niekontrolowane ujawnienie informacji (ang. information leak)
- Atak w architekturze AUTOSAR, odbywa się najczęściej przez:
  - Zakłamanie wskaźnika – ustawienie go na nieprawidłową wartość
  - Przepełnienie/niedopełnienie bufora
  - Nieprawidłowe formatowanie napisu
  - Złe rzutowanie wskaźnika

Model ataku CWE (MITRE)



Wymogi bezpiecznego kodowania

- Obsługa ciągu znaków (ang. string) w języku C, obfituje w wiele niespodzianek
- Niektóre wywołania (strcpy(…) i podobne), z samej natury są niebezpieczne i posiadają swoje bezpieczne odpowiedniki (strncpy(…))
- Środki zaradcze to:
  - Rekomendacje – SEI CERT C/C++ Coding Standards, MISRA C/C++, HIC++, JPL C/C++ Coding Standards, C++ Core Guidelines
  - Narzędzia – CodeSonar, RATS, FlawFinder, compiler sanitization

Code Corrutpion Attack

- Atak polega na złamaniu polityki integralności kodu (CVE)
- Bity zabezpieczające prawa dostępu do strony pamięci (R/W/X), nie mogą mieć dowolnych ustawień
- Kombinacja bezpieczna to W xor X
- Mechanizm ten nazywany jest DEP (ang. Data Execution Prevention)
- Stosy technologiczne języków z JIT (ang. Just-In-Time), mogą łamać tę politykę jawnie

Control-flow Hijack Attack (1/2)

- Metody z tego obszaru, skupiają się na przekłamaniu wskaźnika który:
  - Służy do zapisu funkcji powrotu – metody ROP (ang. Return-Oriented Programming)
  - Służy do wykonania skoku – metody JOP (ang. Jump-Oriented Programming)
  - Powrót do biblioteki libc
- Kod ataku, zapisywany jest najczęściej na stosie w postaci bufora do którego następuje ustawienie rejestru PC i wykonanie
- Obecne kompilatory posiadają narzędzia wdrażające środki zaradcze na tego typu ataki:
  - CPI (ang. Code Pointer Integrity)
  - CPS (ang. Code Pointer Separation)
  - SSP (ang. Stack Smashing Protection)

Control-flow Hijack Attack (2/2)

- Dodatkowo na poziomie systemu operacyjnego, często wdrażana jest polityka randomizacji adresów ASLR (ang. Address Space Layout Randomization)
- Inne ważne techniki:
  - Shadow Stack – instrukcje powrotu zapisywane są na oddzielnym stosie który staje się arbiterem
  - Control-flow integrity (CFI) – dodaje informację o wykonywanych skokach sprawdzając graf wywołań

Atak na dane/wyciek informacji

- Mechanizmy obrony:
  - Data Space Randomization – system przydziela losowo wybrane strony pamięci, ładuje biblioteki współdzielone do nieprzewidzianych (przez proces) obszarów pamięci
  - Write Integrity Testing – badanie sum kontrolnych zapisanych danych w krytycznych momentach działania aplikacji
  - ASLR (ang. Address Space Layout Randomization) – pamięć procesu z każdym jego uruchomieniem ma inną przestrzeń adresową
  - Data Space Randomization – adres pamięci danych z każdym uruchomieniem jest inny

Mechanizmy kompilatora

- Zabezpieczenie stosu (SSP), możliwe jest w wielu kompilatorach po użyciu odpowiednich przełączników wpływających na generowany kod
- Dla gcc / clang:
  - **–fstack-protector** – dodanie na końcu stosu 8 bajtów stażnika
  - **–fstack-protector-all** – dodatkowe zabezpieczenie funkcji
  - **–fstack-protector-strong –** zabezpieczone są także definicje tablic lokalnych i adresy ramki stosu
  - **– fstack-protector-explicit** – zabezpiecza wyłącznie funkcje z atrybutem stack_protect (tylko gcc)

ASLR

- Wszystkie wiodące systemy operacyjne obecnie stosują tę technikę
- Wymaga ona kompilacji z emisją kodu działającego w nieoznaczonej przestrzeni adresowej (Position Independent Execution)
- Dla gcc/clang to przełącznik –fpie a dla bibliotek –fPIC

CFI (ang. Control-flow Integrity)

- Przy stosowaniu tej techniki, wzrasta objętość binariów ~20% a czas wykonania spada o ~2%
- Dla gcc:
  - **-fvtable-verify=[std|preinit|none]** – działa tylko dla C++ i polimorfizmu dynamicznego
- Dla clang:
  - **--fsanitize=[cfi-cast-strict | cfi-derived-cast | cfi-unrelated-cast | cfi-novcall | cfi-vcall | cfi-icall | cfi]** – odpowiednie poziomy zabezpieczeń, cfi uruchamia wszystkie
  - **--fsanitize=safe-stack** – dodanie stosu-cienia, wpływa na objętość wydajność ~10%

Inne mechanizmy

- Przeniesienie katalogu głównego – chroot
- Kontrola użycia zasobów – ulimit
- Przestrzenie nazewnicze – Linux namespace, LXC, Docker, systemd-nspawn
- *BSD – więzienie - Jails