Kod szkolenia: **RE**

Tytuł szkolenia: **Inżynieria odwrotna kodu (Reverse Code Engineering) w systemach Windows i Linux oraz metody zabezpieczania programów**

Dni: 5

Opis:

Adresaci szkolenia

Szkolenie jest adresowane do programistów, testerów oraz specjalistów z zakresu bezpieczeństwa aplikacji.

Cel szkolenia

Celem szkolenia jest przedstawienie technik inżynierii odwrotnej (ang. reverse engineering, RE) aplikacji, protokołów i formatów plików na platformach typu desktop (x86, x86-64, Java, .NET). Istotnym elementem szkolenia są warsztaty podczas których uczestnicy przećwiczą poznane techniki na praktycznych przykładach używając wiodących w branży narzędzi takich jak IDA Pro, OllyDbg, Immunity Debugger i inne. Wiedza zdobyta podczas szkolenia pozwoli na lepszą ochronę tworzonych aplikacji przed stosowaniem inżynierii odwrotnej przez atakujących jak również na obchodzenie niektórych z zabezpieczeń. Uczestnik szkolenia będzie umiał identyfikować formaty plików, algorytmy (w tym algorytmy kryptograficzne) oraz monitorować działanie aplikacji. Szkolenie jest podstawą dla kolejnych szkoleń z zakresu bezpieczeństwa dotyczących analizy malware, inżynierii odwrotnej na platformach mobilnych oraz technik informatyki śledczej i analizy powłamaniowej.

Mocne strony szkolenia

Podczas szkolenia uczestnicy poznają metody reverse engineeringu oraz zabezpieczania oprogramowania przed inżynierią odwrotną pod opieką doświadczonego trenera. Przykłady zostały dobrane tak by mogły się na nim odnaleźć osoby o różnym poziomie doświadczenia.

Wymagania

Od uczestników wymagana jest znajomość podstaw działania komputera na poziomie architektury, umiejętność rozumienia kodu napisanego w językach typu C/C++, Java i .NET oraz biegłe posługiwanie się wybranym systemem operacyjnym: Windows lub Linux. Wskazana jest znajomość jednego z popularnych języków skryptowych a także algorytmów i struktur danych.

Parametry szkolenia

5 * 8 godzin (7 godzin netto) wykładów połączonych z warsztatami i ćwiczeniami (z wyraźną przewagą warsztatów).

Wielkość grupy: 5 - 8 osób

Program szkolenia:

- Wprowadzenie do tematyki reverse engineering
  - co może podlegać inżynierii wstecznej
  - czym jest inżynieria odwrotna oprogramowania
  - legalność inżynierii wstecznej
  - jakie są etapy procesu reverse engineering?
  - zastosowanie reverse engineering: kompatybilność, wyszukiwanie błędów, analiza złośliwego kodu, łamanie zabezpieczeń, odzyskiwanie kodu źródłowego, modyfikacja i poprawianie oprogramowania (reengineering)
- Podstawy asemblera
  - działanie procesora
  - instrukcje
  - proste programy w asemblerze
  - różnice pomiędzy platformami x86 a x86-64
  - asemblery NASM i FASM
  - kompilacja programów w assemblerze
  - śledzenie programów z użyciem debuggera: x64dbg
- Utrudnianie inżynierii odwrotnej - jak się zabezpieczyć?
  - sztuczki “antydebug”: wykrywanie śledzenia i piaskownicy (sandbox)
  - wykrywanie maszyn wirtualnych
  - zaciemnianie znaczenia kodu asemblera (obfuskacja kodu)
  - optymalizacja kodu kompilowanego
  - maszyny pseudowirtualne
  - wykrywanie modyfikacji i pułapek
  - zabezpieczenia warstwowe i emulacja kodu
  - użycie protektorów i pakerów
  - zaciemnianie kodu w językach opartych o maszyny wirtualnego na przykładzie yGuard, Proguard, Excelsior JET
- Etapy procesu RE
  - identyfikacja celu
  - monitorowanie celu
  - usunięcie utrudnień
  - deasemblacja
  - identyfikacja algorytmów i bloków budujących np. funkcji bibliotecznych
  - dekompilacja i rekonstrukcja kodu źródłowego
  - wykorzystanie instrumentacji i profilowania
- Identyfikacja celu
  - detekcja formatów pliku wykonywalnego

    - detekcja pakerów i protektorów
    - identyfikacja ciągów znaków w tym ukrytych ścieżek
    - narzędzia: PEiD, PEbear
- Monitorowanie celu
    - otwieranie plików i ścieżek rejestru
    - komunikacja sieciowa
    - narzędzia: Process Monitor (Regmon+Filemon)
- Usuwanie utrudnień w reverse engineering
    - ręczne i automatyczne usuwanie pakerów i protektorów
    - manualna rekonstrukcja plików wykonywalnych
    - narzędzia: OllyDbg, OllyDump
- Debugowanie
    - techniki debugowania
    - używanie debuggera
    - narzędzia: Immunity Debugger, OllyDbg, WinDbg
    - użycie maszyn wirtualnych i zdalnego debugowania
- Instrumentacja i profilowanie
    - wykorzystanie w celu śledzenia
    - narzędzia: PIN Framework
- Deasemblacja kodu
    - co to jest deasemblacja, trudności deasemblacji
    - proces deasemblacji: IDA Pro
    - identyfikacja funkcji bibliotecznych
        - jak działa mechanizm FLIRT?
        - przygotowywanie sygnatur
    - zastosowania skryptów IDC w IDA Pro
    - skrypty w IDAPython
    - debugger i emulator IDA Pro
- Postępowanie z typowymi celami na platformie Windows
    - pliki natywne (.exe - PE i PE64)
        - format pliku
        - proces ładowania i wykonywania pliku przez system
        - zasoby i edycja zasobów
    - Visual Basic i języki posługujące się bytecode (p-code)
        - dedykowane dekompilatory
    - Delphi i C++ Builder
        - dedykowane narzędzia: DeDe
    - cele egzotyczne: pliki obiektowe, skompilowane pliki skryptów w językach interpretowanych typu Python
- Postępowanie z typowymi celami na platformie Linux
    - pliki natywne - ELF i a.out
    - pliki obiektowe
    - dedykowane narzędzia Linux: GDB, Fenris, IDA Pro etc.
- Postępowanie z programami Java
    - użycie dekompilatorów Java
    - użycie deasemblerów bajtkodu (bytecode) dla Java

    - zabezpieczanie i utrudnianie reverse engineeringu kodu Java
- Postępowanie z programami .NET
    - użycie dekompilatorów - Reflector
    - użycie deasemblerów Intermediate Languange (ildasm)
    - zabezpieczanie i utrudnianie reverse engineeringu kodu .NET
- Charakterystyczne cechy języków C i C++
    - tablice metod wirtualnych
    - mangling
    - konstruktory i destruktory
    - argumenty funkcji i metod
    - zmienne i zmienne rejestrowe
    - informacje symboliczne (symbole debugowania)
    - przykłady dla Visual C++, GCC
    - cele tworzone przez inne kompilatory: Watcom, Borland C/C++
- Wykopki muzealne
    - pliki .com i exe 16-bit
    - Pascal, FoxPro etc.
- Reversing algorytmów
    - obserwacja symptomów działania
    - identyfikacja przepływu danych
    - wyodrębnianie bloków składowych
    - metody identyfikacji: magiczne ciągi, sygnatury metod bibliotecznych, metody heurystyczne
    - identyfikacja algorytmów kryptograficznych
- Reversing protokołów komunikacyjnych i formatów plików
    - wykorzystanie snifferów
    - zautomatyzowany reversing protokołów
    - narzędzia: Hachoir
- Reversing formatów plików
    - typowe zagadnienia spotykane w plikach: sygnatury, sumy kontrolne, kompresja
    - narzędzia: MultiExtractor