
Spis treści

Wstęp	15
1. Funkcje, struktury, klasy i obiekty na niskim poziomie	19
1.1. Wywoływanie funkcji w językach (hardco) niskiego poziomu	21
1.1.1. CALL, RET i konwencje wywołań	21
1.1.2. Konwencje wywołań x86	24
1.1.3. Konwencje wywołań x86-64	27
1.2. Struktury	28
1.2.1. „Zgadywanie” wielkości i składowych elementów struktury w pamięci ...	30
1.2.2. Rozpoznawanie budowy struktur lokalnych i globalnych	30
1.2.3. Rozpoznawanie budowy struktur dynamicznie alokowanych	32
1.3. Klasy, obiekty, dziedziczenie i tablice wirtualne	34
1.3.1. Pusta klasa z strukturą	34
1.3.2. Obiekty = struktury + funkcje + tablice	38
1.3.3. Waryjacje tematów w rodzinie, czyli dziedziczenie	38
1.4. Podsumowanie	40
2. Środowisko uruchomieniowe na systemach GNU/Linux	43
2.1. Wstęp	45
2.2. Pliki wykonywalne ELF	45
2.2.1. Identyfikacja systemu i architektury docelowej	46
2.2.2. Segmenty	46
2.2.3. Segment PT_LOAD	51
2.2.4. Segment PT_DYNAMIC	53
2.2.5. Selekcja dynamic	53
2.2.5.1. Deklaracja bibliotek zależnych	57
2.2.5.2. Wczytanie inicjalizacji programu	59
2.3. Środowisko uruchomieniowe	62
2.3.1. Kład PIC	62
2.3.2. Tablice GOT i PLT	65
2.3.3. Program ładujący klasę	73

2.3.3.1.	Zmiana środowiska	73
2.3.3.2.	LD_LIBRARY_PATH	74
2.3.3.3.	LD_PRELOAD	75
2.3.3.4.	LD_AUDIT	77
2.3.4.	Zmiana pamięci procesu	81
2.3.4.1.	System plików /proc	81
2.3.4.2.	Pliki specjalne w /proc/pid	82
2.3.4.3.	Pliki specjalne maps i mem	83
2.3.4.4.	VSPO	85
2.3.4.5.	Wektory inicjalizacyjne	86
2.3.5.	Wierzytowanie kodu	88
2.3.5.1.	Wersja praca(2)	90
2.3.6.	Samomodyfikujący się kod	96
2.4.	Podsumowanie	98
	Bibliografia	99
3.	Mechanizmy ochrony aplikacji	103
3.1.	Wstęp	103
3.2.	Przepełnienie bufora na stosie	104
3.3.	Procedury obsługi wyjątków	108
3.4.	Zapobieganie wykonaniu danych	112
3.5.	Łatwość ataku przestrzeni adresu	114
3.6.	Dodatkowe materiały	118
3.7.	Podsumowanie	118
	Bibliografia	118
4.	Metody przeciwdziałania odłudzeniu kodu aplikacji z pamięci procesu	119
4.1.	Wstęp	120
4.1.1.	Zawartość rejestra	120
4.2.	Packery, symulatory i protokół plików PE	122
4.3.	Emulowanie zabezpieczeń	126
4.4.	Ochrona przed odłudzeniem kodu programu zapisanego z pamięci procesu	128
4.5.	Naukowcy	130
4.6.	Skrócony punkt rozpoczęcia programu	139
4.7.	Przekierowanie i obfuskacja importowanych funkcji	142
4.8.	Podsumowanie	147
	Bibliografia	148
5.	NET internals – format & PE	151
5.1.	Wstęp	153
5.2.	Format pliku .NET	153
5.2.1.	JIT, czyli drugi sposób kompilacji programu	153
5.2.2.	Język pośredni CLR	154

5.2.1.1.	Przykład 1: <code>add()</code>	155
5.2.1.2.	Przykład 2: <code>stringWriteTest()</code>	155
5.2.1.3.	Przykład 3: <code>programFlowTest()</code>	157
5.2.3.	Dedykowane konstruktory metodowych	161
5.2.4.	Wysokoopinionowana struktura programu zachowana w pliku wykonywalnym	161
5.2.5.	Tokem	164
5.2.6.	Funkcje natywne	165
5.2.7.	AOIT	165
5.3.	Intencja wzmocna	166
5.3.1.	Analiza statyczna	166
5.3.2.	Dekompilacja	166
5.3.3.	Rekompilacja i odtwarzanie działania programu	170
5.3.4.	Modyfikacja istniejących metod	170
5.3.5.	Debug	174
5.3.5.1.	<code>de5py</code>	174
5.3.5.2.	Wyrzki do WinDbg	176
5.4.	Metody protokołu plików .NET	176
5.4.1.	Nadpisywanie nazw nadanych przez użytkownika	176
5.4.2.	Szyfrowanie stałych tekstowych	176
5.4.3.	Utworzenie dekompilecji	178
5.4.4.	Ukrywanie kodu programu	178
5.4.5.	Decompilacja	179
5.5.	Podsumowanie	180
	Bibliografia	180
6.	Python – obfiskacja i intencja wzmocna	181
6.1.	Wstęp	183
6.2.	Obfiskacja a model wykonania	183
6.3.	Obfiskacja źródeł	185
6.4.	<code>PEKi.py</code> i <code>pye</code>	188
6.5.	Bundley i kompilator	190
6.5.1.	<code>PyInc</code>	190
6.5.2.	<code>in_Python</code>	192
6.5.3.	<code>PyInstaller</code>	194
6.5.4.	<code>Nuitka</code>	196
6.5.5.	Inne bundley i kompilatory	205
6.6.	Obfisky-code i function	205
6.7.	Kod bajtowy i gdzie go szukać	213
6.8.	Prosta obfiskacja kodu bajtowego	218
6.9.	Samozanadifikujący się kod bajtowy	222
6.10.	Podsumowanie	224
	Bibliografia	224

7. Maszyna w czarnej skórze, czyli o wstrzykiwaniu kodu w inne procesy	125
7.1. Wstęp	127
7.2. Przegląd technik wstrzykiwania	128
7.2.1. Schemat działania	128
7.2.2. Przygotowania przed implementacją	128
7.2.3. Wybieranie celu	128
7.2.3.1. Wstrzykiwanie kodu do istniejącego procesu	128
7.2.3.2. Wstrzykiwanie kodu do nowego uruchomionego procesu	128
7.2.4. Wypisywanie kodu do obcego procesu	131
7.2.5. Metody przekierowania do wstrzykiwanego kodu	133
7.2.5.1. Uruchomienie docelowego kodu w nowym wątku	133
7.2.5.2. Dodawanie do istniejącego wątku (przy użyciu <i>MyQueueAppThread</i>)	134
7.2.5.3. Nadpisywanie punktu wejścia procesu	135
7.2.5.4. Nadpisywanie kontekstu procesu	138
7.2.5.5. Dodawanie do obiektu <i>Trp</i>	139
7.2.5.6. <i>PowerLoader</i>	141
7.3. Tworzenie wstrzykiwanego kodu	142
7.3.1. Podstawy składowi	142
7.3.2. Warunki wstrzykiwania plików PE	144
7.3.2.1. Wstępne przygotowanie pliku PE	145
7.3.3. Samodzielne ładowanie pliku PE	146
7.3.3.1. Kontrowersja naszego obszaru na wirusach	146
7.3.3.2. Pobieranie adresów tabel	148
7.3.3.3. Rozwijanie relokacji	150
7.3.3.4. Rozwijanie importów	153
7.3.3.5. Przekierowanie wykonania do docelowego pliku PE	161
7.3.3.6. Plik PE o cechach składowi	161
7.3.4. Debugowanie wstrzykiwanego kodu	162
7.4. Metody wstrzykiwania plików PE	164
7.4.1. Klasyczny DLL Injection	165
7.4.1.1. Demonstracja	165
7.4.1.2. Implementacja	166
7.4.2. <i>RanPE</i> (Process Hollowing)	167
7.4.2.1. Demonstracja	167
7.4.2.2. Implementacja	169
7.4.3. <i>ChimeraPE</i>	173
7.4.3.1. Demonstracja	174
7.4.3.2. Implementacja	175
7.4.4. <i>Reflective DLL injection</i>	177
7.4.4.1. Demonstracja	177
7.4.4.2. Implementacja	178
7.4.5. Wstrzykiwanie DLL na poziomie systemu operacyjnego	182

7.4.5.1.	AppInit_DLLs	283
7.4.5.2.	Shell + ImportDll Tag	283
7.5.	Podsumowanie	285
	Bibliografia	286
8.	ELFie brudno szafuści	287
8.1.	Wstęp	289
8.2.	ELFie struktury	289
8.3.	Rozdzielanie jądri	292
8.3.1.	Niewielkiy kod	292
8.4.	Mają: przechwytywać resolver	299
8.5.	Klaszka wlocznie tywa	302
8.7.	DYNAMICResolver zmiany	307
8.7.1.	Co dwie tablice to nie jedna	307
8.7.2.	Jeden bajt by nim rozdzielić	309
8.8.	Podsumowanie	311
	Bibliografia	311
9.	Lamanie zrównoważonych technik przekierowania API	313
9.1.	Wstęp	313
9.2.	Format PE – podstęp	315
9.3.	Tabela importów	319
9.4.	Tabela eksportów	320
9.5.	Prosta metoda obfuskacji IAT	322
9.6.	Przekierowanie API z przepływaniem kodu	323
9.7.	Zupełny to łapież	325
9.8.	Weryfikacja hipotezy pod debuggerem	326
9.9.	Automatyczne przepływanie plików DLL	326
9.10.	Pierwsza próba i pierwsze problemy	330
9.11.	Modyfikacja NTDLL	333
9.12.	Ostatnie poprawki	335
9.13.	Etaki końcowy	336
9.14.	Podsumowanie	337
	Bibliografia	337
10.	Śledzenie śladu wykonania procesu w systemie Linux	339
10.1.	Wstęp	340
10.2.	Metody programowe	342
10.2.1.	Instrumentacja kodu źródłowego	342
10.2.2.	Instrumentacja programu w ramach kompilacji	344
10.2.3.	Instrumentacja kodu binarnego	347
10.2.4.	Śledzenie wywołań systemowych	349
10.2.5.	Śledzenie wywołań funkcji bibliotecznych	350
10.2.6.	Emulacja instrukcji	351

10.2.7. Nagrywanie i odczytywanie próbnego śledztwa wykonania	353
10.3. Metody programowo-sprzętowe	355
10.3.1. Próbkiwanie instrukcji	356
10.3.2. Rejestry debuggera	357
10.3.3. Wykazywanie kroków	360
10.3.4. BTF (Branch Trap Flag)	362
10.3.5. LBR (Intel Last Branch Record)	363
10.3.6. BTS (Intel Branch Trace Store)	365
10.3.7. IPT (Intel Processor Trace)	367
10.4. Metody sprzętowe	370
10.5. Porównanie wydajności	371
10.6. Podsumowanie	372
11. Ciężarstwo zagłady – studium przypadku eksploatacji routera	373
11.1. Wstęp	375
11.2. Kamola	375
11.3. Architektura MIPS	379
11.4. Ciężarstwo i błąd brzoza	381
11.5. Co w tamte plany	383
11.5.1. Funkcja sub_80340E98	385
11.5.2. Funkcja sub_80340F08	389
11.5.3. Funkcja sub_803407D0	394
11.5.4. Argumenty dla funkcji sub_8034280C	396
11.5.5. Funkcja sub_8034280C	398
11.6. Wykorzystanie podmożli	400
11.6.1. Adres docelowy	401
11.7. Podsumowanie	404
Bibliografia	405
12. W pogoni za flagą – eksploatacja na systemach Windows i Linux	407
12.1. Wstęp	409
12.2. Lokalne testowanie rozwiązań	410
12.3. Multipurpose Calculation Machine (średnio trudne)	411
12.4. Memory (łatwe)	414
12.5. Crypto Machine (średnio trudne)	415
12.6. Quarantine (średnio trudne)	444
12.7. Night Sky (bardzo trudne)	461
12.8. Bubblygum (średnio trudne)	480
12.9. Antipasto (łatwe)	501
12.10. Entree (średnio trudne)	511
12.11. Podsumowanie	518
Bibliografia	518
Indeks	520