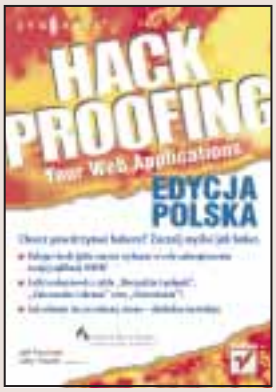
Hack Proofing Your Web Applications. Edycja polska



Autorzy: Jeff Forristal, Julie Traxler
Tłumaczenie: Mikołaj Barwicki
ISBN: 83-7361-006-5
Tytuł oryginału: Hack Proofing Your Web Applications
Format: B5, stron: 406
Przykłady na ftp: 22 kB

Wyczerpujący przewodnik wprowadzający w arkana tworzenia bezpiecznych aplikacji WWW. Twórca aplikacji WWW nie może sobie pozwolić na to, by jego dzieła były wrażliwe na ataki hakerów.

Zacznij więc myśleć jak haker, a luki w bezpieczeństwie natychmiast się ujawnią. Dzięki tej książce nauczysz się analizować metody stosowane do włamań i ataków na witryny WWW. Będziesz mógł następnie wykorzystać tę wiedzę, by zapobiegać atakom.

W książce omówiono m.in.:

- Najlepsze zabezpieczania aplikacji WWW
- Zagadnienia bezpieczeństwa w projektowaniu
- Ostrzeżenia o niebezpieczeństwie
- Technologie XML, Java, ColdFusion oraz skrypty CGI.
- Witryny poświęcone hakerom
- Narzędzia i pułapki
- Pięć faz włamania
- Rodzaje ataków hakerskich
- Niezbędne etapy działania przy ocenie ryzyka
- Automatyczne narzędzia skanujące

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Podziękowania 9

Współpracownicy 10

Przedmowa 13

Rozdział 1. Metodologia włamań 15

Wprowadzenie 15
Podstawowe terminy 16
Krótka historia hackingu 17
Hacking sieci telefonicznych 17
Hacking komputerowy 18
Co motywuje hakera? 20
Hacking etyczny a hacking złośliwy 20
Współpraca ze specjalistami ds. bezpieczeństwa 21
Współczesne rodzaje ataków 22
DoS (DDoS) 22
Hacking wirusowy 24
Kradzież 29
Zagrożenia bezpieczeństwa aplikacji WWW 32
Ukryta manipulacja 32
Zamiana parametrów 33
Zewnętrzne skrypty 33
Przepełnienie buforów 33
Zatrute cookies 34
Zapobieganie włamaniom poprzez przyjęcie postawy hakera 34
Podsumowanie 36
Skrót rozwiązań 37
Pytania i odpowiedzi 39

Rozdział 2. Jak nie zostać „maszynką do kodu”? 41

Wprowadzenie 41
Kim jest „maszynka do kodu”? 42
Postępując zgodnie z zasadami 45
Twórcze kodowanie 46
Zastanawiać się 48
Bezpieczeństwo z punktu widzenia „maszynki do kodu” 50
Programowanie w próżni 51
Tworzenie sprawnych i bezpiecznych aplikacji sieciowych 52
Przecież mój kod działa! 56
Podsumowanie 61
Skrót rozwiązań 62
Pytania i odpowiedzi 63

Rozdział 3. Ryzyko związane z kodem przenośnym 65
Wprowadzenie 65
Działanie ataków wykorzystujących kod przenośny 66
Ataki z wykorzystaniem przeglądarek 66
Ataki z wykorzystaniem programów pocztowych 67
Złośliwe skrypty lub makra 68
Identyfikacja popularnych form kodu przenośnego 68
Języki makropoleceń: Visual Basic for Applications (VBA) 69
JavaScript 74
VBScript 77
Aplety w Javie 79
Kontrolki ActiveX 82
Załączniki listów elektronicznych i pobrane pliki wykonywalne 86
Ochrona systemu przez atakami wykorzystującymi kod przenośny 89
Aplikacje bezpieczeństwa 90
Narzędzia na stronach WWW 93
Podsumowanie 93
Skrót rozwiązań 95
Pytania i odpowiedzi 95

Rozdział 4. Wrażliwe skrypty CGI 97
Wprowadzenie 97
Czym jest i co robi skrypt CGI? 98
Typowe zastosowania skryptów CGI 99
Kiedy powinno się stosować CGI? 103
Zagadnienia związane z instalacją skryptów CGI 104
Włamania umożliwione przez słabe skrypty CGI 105
Jak pisać bardziej szczelne skrypty CGI? 106
Polecenia katalogów 108
Opakowania skryptów CGI 109
Języki wykorzystywane do tworzenia skryptów CGI 112
Powłoka systemu Unix 113
Perl 113
C (C++) 114
Visual Basic 114
Korzyści ze stosowania skryptów CGI 115
Zasady tworzenia bezpiecznych skryptów CGI 115
Składowanie skryptów CGI 118
Podsumowanie 120
Skrót rozwiązań 120
Pytania i odpowiedzi 123

Rozdział 5. Techniki i narzędzia włamań 125
Wprowadzenie 125
Cele hakera 126
Omijanie alarmów 127
Zdobywanie dostępu 128
Zniszczenia, zniszczenia, zniszczenia 130
Jak być lepszym od hakera 131
Pięć etapów włamania 132
Tworzenie planu ataku 132
Tworzenie planu przebiegu włamania 134
Wybranie punktu wejścia 135
Stały i szeroki dostęp 136
Atak 137

Socjotechnika ... 138
Informacje poufne ... 138
Celowo pozostawione tylne wejścia ... 143
Wprowadzenie do kodu ukrytego hasła ... 143
Wykorzystanie słabych stron właściwych kodowi lub środowisku programowania ... 145
Narzędzia wykorzystywane przez hakera ... 145
Edytory szesnastkowe ... 145
Programy uruchomieniowe ... 147
Deasemblery ... 148
Podsumowanie ... 150
Skrót rozwiązań ... 150
Pytania i odpowiedzi ... 153

Rozdział 6. Kontrola kodu i odwrotna analiza ... 155
Wprowadzenie ... 155
Jak efektywnie śledzić działanie programu ... 156
Monitorowanie i kontrola w wybranych językach programowania ... 158
Java ... 158
Java Server Pages ... 159
Active Server Pages ... 159
Server Side Includes ... 159
Python ... 159
Tool Command Language ... 160
Practical Extraction and Reporting Language ... 160
PHP: Hypertext Preprocessor ... 160
C (C++) ... 160
ColdFusion ... 161
Lokalizacja słabych punktów ... 161
Pobieranie danych od użytkownika ... 161
Lokalizacja potencjalnych błędów przepełnienia buforów ... 162
Weryfikacja wyświetlanych informacji ... 165
Kontrola operacji na systemie plików ... 168
Uruchamianie zewnętrznego kodu i programów ... 170
Zapytania do baz danych SQL ... 172
Sieci i strumienie komunikacyjne ... 174
Praktyka ... 175
Podsumowanie ... 176
Skrót rozwiązań ... 176
Pytania i odpowiedzi ... 177

Rozdział 7. Bezpieczeństwo w języku Java ... 179
Wprowadzenie ... 179
Architektura bezpieczeństwa Javy ... 180
Model bezpieczeństwa w języku Java ... 181
Piaskownica ... 183
Podejście do problemów bezpieczeństwa w Javie ... 187
Programy ładujące klasy ... 187
Weryfikator bajt-kodu ... 190
Chronione domeny Javy ... 194
Potencjalne luki bezpieczeństwa w Javie ... 201
Ataki DoS (degradacji usług) ... 202
Konie trojańskie ... 204
Programowanie funkcjonalnych, ale bezpiecznych apletów w Javie ... 205
Skróty wiadomości ... 206
Podpisy cyfrowe ... 208

Uwierzytelnianie ... 214
Ochrona zabezpieczeń za pomocą podpisywania plików JAR ... 220
Szyfrowanie ... 223
Zalecenia Sun Microsystems odnośnie bezpieczeństwa w Javie ... 227
Podsumowanie ... 230
Skrót rozwiązań ... 231
Pytania i odpowiedzi ... 232

Rozdział 8. Bezpieczeństwo w języku XML ... 235
Wprowadzenie ... 235
Definicja języka XML ... 236
Struktura logiczna ... 237
Elementy ... 237
Dokumenty XML, XSL i DTD ... 240
Zastosowanie szablonów w języku XSL ... 240
Zastosowanie wzorów w języku XSL ... 241
DTD ... 243
Wykorzystanie języka XML do tworzenia aplikacji WWW ... 245
Ryzyko związane z językiem XML ... 247
Problemy tajności ... 248
Bezpieczeństwo w języku XML ... 249
Specyfikacja XML Encryption ... 250
Specyfikacja XML Digital Signatures ... 254
Podsumowanie ... 256
Skrót rozwiązań ... 257
Pytania i odpowiedzi ... 258

Rozdział 9. Tworzenie bezpiecznych sieciowych kontrolek ActiveX ... 259
Wprowadzenie ... 259
Zagrożenia związane z technologią ActiveX ... 260
Unikanie typowych luk bezpieczeństwa w kontrolkach ActiveX ... 262
Usuwanie skutków luk w technologii ActiveX ... 264
Metodologia tworzenia bezpiecznych kontrolek ActiveX ... 267
Parametry bezpieczeństwa obiektu ... 267
Bezpieczne kontrolki ActiveX ... 268
Podpisywanie kontrolek ... 268
Znakowanie kontrolek ... 271
Podsumowanie ... 276
Skrót rozwiązań ... 276
Pytania i odpowiedzi ... 278

Rozdział 10. Bezpieczeństwo w ColdFusion ... 281
Wprowadzenie ... 281
Jak działa ColdFusion? ... 282
Zalety szybkiego projektowania ... 283
Zarys znacznikowego języka ColdFusion ... 284
Zachowanie bezpieczeństwa w technologii ColdFusion ... 286
Projektowanie z uwzględnieniem bezpieczeństwa ... 288
Bezpieczne rozpowszechnianie ... 297
Przetwarzanie w aplikacjach ColdFusion ... 297
Sprawdzanie istnienia danych ... 298
Kontrola typów danych ... 299
Szacowanie danych ... 301
Ryzyko związane z technologią ColdFusion ... 302
Zastosowanie programów obsługi błędów ... 304

Stosowanie śledzenia sesji ... 308
Podsumowanie ... 309
Skrót rozwiązań ... 310
Pytania i odpowiedzi ... 311

Rozdział 11. Projektowanie aplikacji spełniających wymogi bezpieczeństwa ... 313
Wprowadzenie ... 313
Zalety stosowania aplikacji spełniających wymogi bezpieczeństwa ... 314
Rodzaje zabezpieczeń stosowanych w aplikacjach ... 315
Podpisy elektroniczne ... 316
Pretty Good Privacy ... 317
Protokół S/MIME ... 319
Secure Sockets Layer ... 319
Certyfikaty cyfrowe ... 323
Podstawowe informacje na temat PKI ... 325
Usługi certyfikujące ... 327
Zastosowanie PKI do zabezpieczania aplikacji WWW ... 328
Implementacja PKI w infrastrukturze WWW ... 329
Microsoft Certificate Services ... 330
Netscape Certificate Server ... 333
PKI dla Serwera Apache ... 339
PKI i Secure Software Toolkits ... 341
Testowanie zabezpieczeń ... 341
Podsumowanie ... 343
Skrót rozwiązań ... 345
Pytania i odpowiedzi ... 347

Rozdział 12. Od początku do końca — praca z planem bezpieczeństwa ... 349
Wprowadzenie ... 349
Analiza kodu ... 350
Sprawdzanie kodu ... 351
Perspektywa współpracownika ... 352
Świadomość słabych punktów kodu ... 354
Testy, testy, testy ... 355
Rozsądek podczas kodowania ... 357
Planowanie ... 357
Standardy kodowania ... 358
Narzędzia ... 359
Tworzenie planu bezpieczeństwa ... 362
Planowanie bezpieczeństwa na poziomie sieci ... 364
Planowanie bezpieczeństwa na poziomie aplikacji ... 364
Planowanie bezpieczeństwa na poziomie biura ... 365
Proces bezpieczeństwa aplikacji WWW ... 365
Podsumowanie ... 367
Skrót rozwiązań ... 368
Pytania i odpowiedzi ... 369

Dodatek A Skrót zagadnień omawianych w książce ... 371

Skorowidz ... 389

Rozdział 9.

Tworzenie bezpiecznych sieciowych kontrolek ActiveX

Rozwiązania omawiane w niniejszym rozdziale:

- Zagrożenia związane z technologią ActiveX.
- Metodologia tworzenia bezpiecznych kontrolek ActiveX.
- Bezpieczne kontrolki ActiveX.
- Podsumowanie.
- Skrót rozwiązań.
- Pytania i odpowiedzi.

Wprowadzenie

Kontrolki ActiveX są opracowaną przez Microsoft implementacją standardu *Component Object Model* (*COM*). Firma stworzyła technologię ActiveX w miejsce przestarzałego modelu *Object Linking and Embedding* (*OLE*), stosowanego we wcześniejszych wersjach systemu Windows. Do ulepszeń ActiveX względem modelu OLE należy rozszerzalność modelu i możliwość przetwarzania rozproszonego (DCOM), jak również lepsza wydajność w lokalnych aplikacjach. Kontrolki ActiveX z reguły są tworzone w Visual Basicu lub C++.

Kontrolki ActiveX są w systemie Windows łatwo zauważane i dodają aplikacjom, zwłaszcza WWW, wiele możliwości interakcji. Dobrze komponują się w dokumenty HTML, dzięki czemu bez trudu można je przenosić pomiędzy systemami. Kontrolki ActiveX w aplikacjach mogą służyć do wykonywania powtarzających się czynności lub do wywoływania innych kontrolek, odpowiedzialnych za konkretne operacje. Po instalacji kontrolki ActiveX, jest ona uruchamiana automatycznie i nie musi być ponownie instalowana. Tak naprawdę kontrolka ActiveX może zostać pobrana z odległego serwera za pośrednictwem adresu URL, po czym uruchamiana bez konieczności powtórnego pobierania. Pozwala to na uruchamianie kontrolek ActiveX z poziomu stron WWW.

Problemy bezpieczeństwa dotyczące kontrolek ActiveX są ściśle związane z właściwościami technologii ActiveX. Kontrolki nie są uruchamiane w ograniczonej przestrzeni lub piaskownicy, jak ma to miejsce w przypadku apletów Javy, tak więc stanowią potencjalnie większe zagrożenie dla aplikacji. Ponadto kontrolki ActiveX są zdolne do wszelkich operacji, które może wykonać użytkownik, to znaczy, że są w stanie dodawać i usuwać dane lub zmieniać właściwości obiektów. Wydaje się, iż JavaScript oraz aplety w Javie zdominowały środowisko projektantów internetowych, ale wiele witryn i aplikacji wciąż korzysta z kontrolek ActiveX.

Wciąż pojawiające się wiadomości o włamaniach na witryny internetowe świadczą o tym, że wielu programistów jeszcze nie opanowało sztuki zabezpieczania swoich kontrolek, a przecież ActiveX jest dość dobrze znaną technologią. Niniejszy rozdział ma za zadanie pomóc w identyfikacji i rozwiązywaniu niektórych problemów bezpieczeństwa wynikających ze źle napisanych kontrolek ActiveX (spośród których wiele jest za darmo dostępnych w internecie). Obalone zostaną popularne, błędne twierdzenia dotyczące ActiveX, natomiast wprowadzone będą skuteczne zasady tworzenia bezpiecznych i funkcjonalnych kontrolek ActiveX.

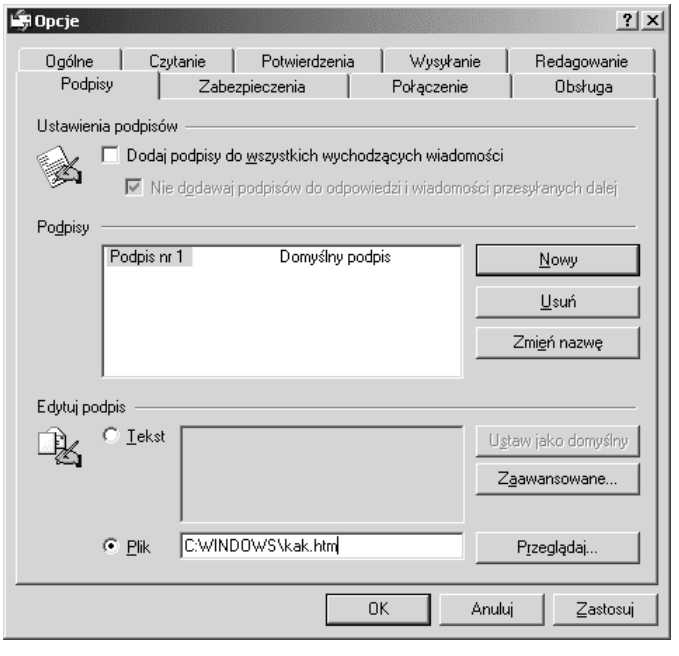
Zagrożenia związane z technologią ActiveX

Podstawowe zagrożenia związane z kontrolkami ActiveX wynikają ze sposobu traktowania problemów bezpieczeństwa przez firmę Microsoft. Wdrażając technologię Authenticode, służącą do opatrywania kontrolki cyfrowym podpisem, Microsoft uznaje, że jest w stanie zagwarantować użytkownikowi źródło pochodzenia kontrolki oraz fakt, że kontrolka nie została zmodyfikowana od momentu powstania. W większości przypadków jest to prawdą, jednak kilka rzeczy, których Microsoft **nie** robi, stanowi poważne zagrożenie dla stacji roboczych i całych sieci. Pierwszym i najbardziej oczywistym niebezpieczeństwem jest fakt, że Microsoft nie ogranicza uprawnień kontrolki po zainstalowaniu jej w lokalnym komputerze. To jedna z kluczowych różnic pomiędzy ActiveX a Javą. Java korzysta z metody zwanej *piaskownicą* (*sandbox*). Umieszczenie apletu Javy w piaskownicy gwarantuje, że aplikacja funkcjonuje we własnym, chronionym obszarze pamięci, który jest odizolowany od innych rzeczy (systemu plików czy innych aplikacji). Nakłada to poważne ograniczenia na możliwości kontrolki. Z kolei kontrolki ActiveX mają takie same prawa jak użytkownik, który je uruchamia po instalacji. Microsoft nie jest w stanie zagwarantować, że z kontrolki korzysta jej autor ani też że jest wykorzystywana zgodnie z przeznaczeniem — na witrynie lub stronach, na potrzeby których powstała. Microsoft nie gwarantuje również, że właściciel witryny lub ktokolwiek inny nie zmienił zawartości strony po tym, jak zainstalowano kontrolkę. To wykorzystanie tych słabych punktów stanowi największe niebezpieczeństwo związane z kontrolkami ActiveX.

Na przykład Scriptlet.Typelib jest kontrolką ActiveX firmy Microsoft, którą projektanci stosują do generacji bibliotek czcionek dla komponentów typu *Windows Script Component* (*WSC*). Jedna z funkcji tej kontrolki pozwala na tworzenie i modyfikację plików w komputerze lokalnym. Oczywiście sprawia to, że kontrolka ta powinna być chroniona przed niepowołanymi programami. Zgodnie z centrum koordynacyjnym CERT (CERT/CC)

kontrolka ta została błędnie oznakowana jako *bezpieczna ze względu na skrypty* i dołączona do pakietów Internet Explorer 4.0 i 5.0. W efekcie haker mógłby napisać fragment złośliwego kodu odwołującego się i uruchamiającego tę kontrolkę bez wiedzy użytkownika. Dwa dobrze znane wirusy wykorzystują tę lukę. Są to: *kak* oraz *BubbleBoy*. Obydwa są roznoszone przez listy elektroniczne HTML, a modyfikują *Rejestr* systemu i inne pliki systemowe. Microsoft w 1999 r. wprowadził poprawkę uodparniającą na te wirusy.

Ponieważ Scriptlet.Typelib oznaczony jest jako *bezpieczny ze względu na skrypty*, domyślne ustawienia programów Internet Explorer, Outlook oraz Outlook Express pozwalają na użycie kontrolki bez zgłaszania alarmów bezpieczeństwa. Lukę tę wykorzystuje wirus kak, który w katalogu startowym Windows usiłuje umieścić plik aplikacji HTML (HTA). Umieszczony tam kak oczekuje na kolejne uruchomienie systemu bądź rejestrowanie użytkownika. Gdy to nastąpi, wirus budzi się i może wyrządzać szkody — wykonuje serię zapisów i modyfikacji różnych plików. W wyniku tego w komputerze pojawia się nowy plik z podpisem, który zawiera wirus i jest dołączany do każdej wychodzącej wiadomości (rysunek 9.1). Dzięki temu kak może się rozprzestrzeniać.



Rysunek 9.1. *Okno opcji programu Microsoft Outlook Express*

Główne uderzenie następuje na skutek sprawdzenia dnia miesiąca i bieżącej godziny. Jeśli jest to godzina 6^{00} po południu lub później, pierwszego dnia miesiąca, kak wyświetla okno z komunikatem *Not Today* (*Nie dzisiaj*, rysunek 9.2). Gdy okno to zostaje zamknięte, kak wywołuje funkcję Win32 API powodującą zamknięcie systemu Windows. Ponieważ kod ten znajduje się w pliku HTA, który jest uruchamiany przy każdym rozruchu i rejestrowaniu, ponowne włączenie zarażonego komputera po godzinie 18^{00} pierwszego dnia miesiąca kończy się wyświetleniem komunikatu *Not Today*, a następnie zamknięciem systemu. Biorąc jeszcze pod uwagę zdolność do tworzenia i modyfikacji plików oraz wpisów rejestru, a także możliwość wywoływania funkcji API, widać, jak ta kontrolka może być niebezpieczna.

Rysunek 9.2.
Okno dialogowe aplikacji HTML

Unikanie typowych luk bezpieczeństwa w kontrolkach ActiveX

Jedna z najpowszechniejszych luk bezpieczeństwa spowodowanych przez technologię ActiveX ma związek z postrzeganiem (lub jego brakiem) przez programistę możliwości kontrolki. Każdy programista zatrudniony w przedsiębiorstwie czy firmie konsultingowej, który pisze kontrolkę na potrzeby legalnej działalności, chce, by kontrolka była jak najłatwiejsza w użyciu. Bierze pod uwagę docelowe zastosowanie kontrolki i jeśli wydaje się bez zarzutu, to oznacza ją jako „bezpieczną ze względu na skrypty". Z drugiej strony bez oznaczenia kontrolki jako bezpiecznej, użytkownik jest zasypywany ostrzeżeniami o potencjalnym zagrożeniu związanym z użyciem kontrolki, która nie jest podpisana i nie jest oznaczona jako bezpieczna. W zależności od ustawień opcji bezpieczeństwa w przeglądarce użytkownik może w ogóle nie móc uruchomić kontrolki (rysunek 9.3). Po oznaczeniu jako bezpiecznej, inne aplikacje i kontrolki mają możliwość uruchomienia jej bez pytania o zgodę użytkownika. Oczywiście taka sytuacja może być niebezpieczna. Dobrym przykładem potencjalnych efektów działania technologii ActiveX jest niesławna kontrolka Windows Exploder. Była to mała, zgrabna kontrolka, stworzona przez Freda McLaina (*www.halcyon.com/mclain/ActiveX*), która miała demonstrować możliwości „niebezpiecznej" technologii. Jedyne, co robiła ta kontrolka, to zamknięcie i wyłączenie zarażonego systemu. Teraz nie wydaje się to zbyt groźne, a nie był to efekt pomyłki, ale wywołuje chwilę zastanowienia. Kontrolki ActiveX wymagają ostrożności. Przed wprowadzeniem jej na rynek programista musi wiedzieć, do czego zdolna jest kontrolka.

Rysunek 9.3.
Alarm Microsoft Internet Explorera

Kolejnym problemem wynikającym z nieostrożności programisty jest wykorzystanie kontrolki niezgodnie z przeznaczeniem, przy wykorzystaniu przywilejów użytkownika. Fakt, że twórca tworzy kontrolkę z myślą o konkretnym przeznaczeniu, nie oznacza, że kto inny nie jest w stanie znaleźć innego zastosowania. Zawsze znajdzie się ktoś, kto zechce spożytkować twórczość programisty. Wystarczy przypomnieć sobie przykład kontrolki Scriptlet.Typelib z poprzedniego podrozdziału. Programiści Microsoftu wiedzieli, że kontrolka prawidłowo tworzy biblioteki czcionek dla WSC, ale nie wzięli pod uwagę, że ktoś mógłby skorzystać z ich kontrolki do utworzenia aplikacji HTML i modyfikacji wpisów rejestru.

Innym problemem związanym z kontrolkami ActiveX jest publikacja wersji, które nie zostały dokładnie przetestowane i zawierają błędy. Wśród błędów często spotykanych w programach w C++ jest błąd *przepełnienia buforów*. Ma on miejsce podczas kopiowania ciągu znaków do tablicy o stałej długości, gdy ciąg jest dłuższy niż tablica. Wynikiem jest przepełnienie buforów i potencjalnie upadek aplikacji. W przypadku tego

rodzaju błędów najistotniejsze jest to, że efekty są nieprzewidywalne. Przy odrobinie szczęścia, pojawi się jedynie szczegółowy komunikat o błędzie (rysunek 9.4). Przepełnienie bufora może spowodować wyświetlenie na ekranie niepożądanych znaków albo spowodować błąd przeglądarki i zawiesić system. Problem ten przez lata był plagą środowiska systemów UNIX (Linux), ale ostatnio coraz bardziej zauważa się go na platformie Windows. W serwisie Microsoft TechNet (*www.microsoft.com/technet/security*) wśród najpopularniejszych zagadnień związanych z bezpieczeństwem w technologiach IT miesięcznie można znaleźć co najmniej jedną wzmiankę na temat tego błędu. Nie jest to wyłącznie problem Microsoftu — dotyczy niemal każdego producenta tworzącego kod dla platformy Windows. Dla ilustracji zasięgu tego problemu warto zajrzeć do niedawnego raportu na witrynie secureroot (*www.secureroot.com*), w którym Neal Krawetz donosi, iż udało mu się zlokalizować błąd przepełnienia buforów w module Shockwave Flash. Raport informuje: „Na witrynie firmy Macromedia można się dowiedzieć, że moduł zainstalowany jest już w 90% wszystkich przeglądarek internetowych. Ponieważ przepełnienie buforów pozwala na uruchomienie dowolnego kodu, to może dotyczyć 90% podłączonych do sieci systemów". To przerażająca wizja! Mimo że ten błąd jest bardzo rozpowszechniony, rozwiązanie jest proste — należy więcej czasu poświęcić na dokładne testowanie i sprawdzenie, czy kod zapewnia kontrolę brzegów dla wszystkich wartości dopuszczających dane o zmiennej długości.

Rysunek 9.4. *Okno dialogowe szczegółowego komunikatu o błędzie w Windows*

Kolejne niebezpieczeństwo wiąże się ze stosowaniem starych, wycofanych już wersji kontrolek ActiveX. Niektóre z nich zawierają błędy, inne nie. Niektóre mogły zostać zupełnie przebudowane lub z jakiegoś powodu zastąpione. W momencie gdy ktoś posiada kopię kontrolki, nie ma gwarancji, że użyje obecnej wersji, zwłaszcza jeżeli może być ona w jakiś sposób nadużyta. W przypadku instalacji kontrolki, której podpis jest przestarzały, wyświetlany jest komunikat o błędzie, jednak wiele osób i tak ją zainstaluje, ponieważ wciąż widnieje na niej podpis jej twórcy (rysunek 9.5). Niestety, nie ma sposobu, by zapobiec wykorzystaniu przez kogoś kontrolki, mimo że twórca ją wycofał. Gdy programista opatruje podpisem i publikuje w internecie kontrolkę, która może być potencjalnie niebezpieczna, kontrolka ta staje się łakomym kąskiem dla każdego hakera.

Rysunek 9.5. *Ostrzeżenie o przestarzałym podpisie kontrolki*

W tym przypadku najlepszą obroną jest skuteczny atak. Dokładne testowanie kontrolki przed opublikowaniem oszczędzi później kłopotów.

Użytkownik również powinien „atakować". Nigdy nie należy instalować kontrolki, która jest niepodpisana lub okres ważności podpisu dobiegł końca. Skutki mogą być potencjalnie groźne. Po instalacji kontrolki ActiveX mają takie same prawa jak użytkownik i mogą wykonywać takie same operacje. Mogą wszystko, od wysłania poufnych danych w załączniku listu elektronicznego do wywołania polecenia powłoki, takiego jak np. *Delete*. Przed podjęciem decyzji o instalacji niepodpisanej lub przestarzałej kontrolki należy zdać sobie sprawę z ryzyka.

Usuwanie skutków luk w technologii ActiveX

Słabość technologii ActiveX jest poważnym problemem dla administratorów sieci, użytkowników i programistów. Dla niektórych skutki niepoprawnego stosowania i zarządzania kontrolkami ActiveX mogą być ogromne. Inni nigdy nie martwili się tym problemem. Istnieje możliwość ustanowienia polityki, która zabroni stosowania wszelkich kontrolek i skryptów, ale trzeba tego dokonać na poziomie każdej maszyny z osobna, a to zabiera dużo czasu, zarówno jeśli chodzi o implementację, jak i zarządzanie. Występuje to zwłaszcza w środowiskach, w których użytkownicy sami chętnie zmieniają parametry przeglądarek. Do innych opcji można zaliczyć ograniczenie dostępu kontrolek ActiveX, na przykład za pośrednictwem barier firewall i oprogramowania antywirusowego, jednak efektywność takich rozwiązań ogranicza się do przypadków oczywistych i znanych. Mimo że pełna ochrona przed zagrożeniami spowodowanymi przez słabości technologii ActiveX jest trudna — jeśli nie niemożliwa — to użytkownicy na każdym poziomie mogą podjąć kroki prowadzące do minimalizacji niebezpieczeństw.

Ochrona na poziomie sieciowym

Administrator sieci powinien rozpocząć dostrajanie różnych parametrów bezpieczeństwa od sieciowego systemu operacyjnego:

- Do nałożenia ograniczeń na kontrolki mogą służyć takie mechanizmy, jak strefy bezpieczeństwa czy protokół Secure Sockets Layer (SSL).

- W rejestrze systemowym dostępny jest wpis `CodeBaseSearchPath` określający miejsce, z którego system będzie próbował pobierać kontrolki ActiveX.
- Istnieje narzędzie o nazwie *Internet Explorer Administration Kit* (*IEAK*), które może służyć do definiowania i dynamicznego zarządzania kontrolkami ActiveX.

Mimo że część tych opcji są bardzo przydatna, należy również rozważyć instalację bariery firewall. Niektóre zapory mają możliwość monitorowania i wybiórczego filtrowania wywołań oraz transferów kontrolek ActiveX. Inne nie oferują takich opcji, dlatego powinno się zbadać możliwości wybieranej bariery firewall.

Ochrona na poziomie klienta

Jedną z najistotniejszych rzeczy, które może wykonać użytkownik, jest ciągła aktualizacja systemu operacyjnego wraz z jego komponentami i oprogramowaniem antywirusowym. Należy regularnie pobierać bieżące poprawki i bazy danych wirusów. Kolejną opcją dla użytkowników, a także administratorów, są parametry stref bezpieczeństwa w programach Internet Explorer, Outlook oraz Outlook Express. Są to cenne narzędzia bezpieczeństwa i warto skorzystać z ich potencjału.

Ustawianie stref bezpieczeństwa

Prawidłowo ustawione strefy bezpieczeństwa mogą obniżyć poziom ryzyka związany z kontrolkami ActiveX. Istnieje pięć stref bezpieczeństwa: *Lokalny intranet*, *Zaufane witryny*, *Witryny z ograniczeniami*, *Internet* oraz *Mój komputer*. Ostatnia strefa, *Mój komputer*, jest dostępna jedynie za pośrednictwem IEAK, nie z poziomu przeglądarki. W przypadku braku dostępu do pakietu IEAK, można zmodyfikować parametry stref bezpieczeństwa poprzez klucz rejestru o nazwie `[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones]`. Odpowiednie wartości tego klucza znajdują się w tabeli 9.1.

Tabela 9.1. *Parametry stref bezpieczeństwa dla programów Internet Explorer, Outlook oraz Outlook Express*

Wartość klucza rejestru	Strefa bezpieczeństwa
0	Mój komputer
1	Lokalny intranet
2	Zaufane witryny
3	Internet
4	Witryny z ograniczeniami

Modyfikacja ustawień stref bezpieczeństwa w programie Internet Explorer 5.x może się odbyć według następującego schematu:

1. Z menu *Narzędzia* wybiera się *Opcje internetowe*. Pojawia się okno dialogowe *Opcji internetowych*.
2. Należy wybrać zakładkę *Zabezpieczenia*. Pojawia się panel opcji zabezpieczeń.

3. Trzeba wybrać strefę do modyfikacji. Dla większości użytkowników będzie to strefa *Internet*, ale w zależności od okoliczności być może konieczne będzie powtórzenie tych kroków dla strefy *Lokalny intranet*.
4. Należy kliknąć przycisk *Poziom niestandardowy*. Pojawi się panel *Ustawienia zabezpieczeń*.
5. Zmiana jednego lub więcej spośród znajdujących się poniżej ustawień pozwoli osiągnąć wymagany poziom bezpieczeństwa:
   - w opcjach *Formanty ActiveX i dodatki plug-in* należy wybrać *włącz*, *wyłącz* lub *monituj*.
   - w opcjach *Wykonywanie skryptów formantów ActiveX oznaczonych jako bezpieczne* należy wybrać *wyłącz* lub *monituj*.
6. Kliknięcie *OK* zatwierdzi te zmiany. Pojawi się okno dialogowe potwierdzające dokonanie zmian.
7. Należy kliknąć *Tak*.
8. Przycisk *OK* zamknie okno *Opcji internetowych* i zapisze ustawienia.

Użytkownik powinien również wyrobić w sobie nawyk ostrożności w momencie, gdy otrzymuje monit dotyczący pobrania lub uruchomienia kontrolki ActiveX. Trzeba również wyłączyć kontrolki ActiveX oraz inne języki skryptowe w aplikacjach pocztowych, co często bywa pomijane. Wiele osób uważa, że skoro nie używają aplikacji pocztowych Microsoftu, to są bezpieczni. Ale jeśli aplikacja pocztowa potrafi wyświetlać strony HTML, to zagrożenie ze strony Eudory jest równie prawdopodobne, jak ze strony programu Outlook Express. W przeglądarkach Netscape na dzień dzisiejszy niebezpieczeństwo związane z ActiveX jest niewielkie. Aby uruchomić kontrolkę ActiveX w przeglądarce Netscape wcześniejszej niż wersja 6. trzeba zainstalować moduł rozszerzający. Najbardziej znanym modułem obsługującym ActiveX w przeglądarkach Netscape jest ScriptActive firmy NCompass (NCompass nie oferuje już jednak tego modułu ani nie obsługuje przeglądarki Netscape). Nie istnieje standardowy moduł dla Netscape 6 z jądrem Gecko. Uruchomionych jest jednak kilka projektów mających na celu stworzenie nowych modułów lub bezpośredniej obsługi technologii ActiveX przez Netscape.

Na programiście spoczywa największa odpowiedzialność. Stanowi on pierwszą linię obrony przed słabościami modelu ActiveX. Trzeba znać narzędzia dostępne w danej chwili, które pozwolą podnieść poziom bezpieczeństwa i zawsze należy brać pod uwagę ryzyko związane z tworzeniem kodu przenośnego. Powinno się stosować do dobrych reguł programowania i zachować szczególną ostrożność, by uniknąć powszechnych problemów i pomyłek otwierających drogę do nadużyć. Przede wszystkim ważna jest dobra ocena i rozsądek oraz testowanie, testowanie, testowanie… Po podpisaniu i opublikowaniu kontrolki staje się ona celem — każdy może jej użyć. Dlatego trzeba mieć pewność, że stworzona kontrolka ActiveX jest tak bezpieczna jak to tylko możliwe.

Hakerzy z reguły wykorzystują pomysłowe sposoby, by skłonić użytkownika do kliknięcia pozornie bezpiecznego odnośnika lub otwarcia listu opatrzonego tytułem, np. „W odpowiedzi na pański komentarz”.

Metodologia tworzenia bezpiecznych kontrolek ActiveX

Jak pisać bezpieczne kontrolki ActiveX? Cóż, jak zawsze, pierwszy krok wymaga trafnej oceny i zdrowego rozsądku. Należy się upewnić, że działanie kontrolki i jej możliwości są dokładnie poznane. Dobre reguły inżynierii programowania oraz techniki projektowe również mogą być pomocne:

- **Trzeba dokładnie udokumentować kontrolkę.** To da administratorom i użytkownikom podstawę do prawidłowej oceny potencjalnego ryzyka związanego z kontrolką.
- **Kontrolka powinna zawierać jak najmniejszą funkcjonalność — konieczną do spełnienia swojego zadania.** Każda dodatkowa funkcja aż prosi się o nadużycie.
- **Należy zwrócić szczególną uwagę na powszechne pomyłki, jak przepełnienie buforów czy błędy poprawności danych wejściowych.**
- **Istotne są metody zabezpieczania kontrolek ActiveX za pomocą odpowiednich parametrów bezpieczeństwa obiektu.**

Parametry bezpieczeństwa obiektu

Zagadnienia bezpieczeństwa obiektów można podzielić na dwie grupy: bezpieczeństwo ze względu na inicjalizację oraz bezpieczeństwo ze względu na skrypty. Według definicji Microsoftu *bezpieczny ze względu na inicjalizację* oznacza, że kontrolka może pobierać dowolne argumenty, zaś *bezpieczny ze względu na skrypty* — każde zastosowanie właściwości, metod i zdarzeń kontrolki jest bezpieczne. Pamiętając o tym, należy dokładnie testować kontrolki i upewnić się, że nie wykonują one potencjalnie niebezpiecznych zadań. Przykładami czynności uważanych za niebezpieczne są tworzenie i usuwanie plików, udostępnianie haseł, podgląd prywatnych informacji o użytkowniku czy wysyłanie zapytań SQL. Mimo że *Lista zagadnień bezpieczeństwa Kontrolek ActiveX* Microsoftu może wydawać się subiektywna, właściwa ocena i rozsądek powinny pomóc w zakwalifikowaniu operacji. Jeśli kontrolka wykonuje **jakąkolwiek** operację spośród wymienionych poniżej, **nie** powinna być oznaczona jako bezpieczna:

- Dostęp do informacji o komputerze lokalnym lub użytkowniku.
- Udostępnianie prywatnych informacji w komputerze lokalnym bądź w sieci.
- Modyfikacja lub usuwanie informacji z komputera lokalnego bądź sieci.
- Błędy kontrolki potencjalnie prowadzące do zakończenia działania przeglądarki.
- Nadmierne zużycie czasu i zasobów, takich jak pamięć.
- Wykonywanie potencjalnie destrukcyjnych wywołań systemowych (włącznie z uruchamianiem plików).
- Podstępne wykorzystanie kontrolki i powodowanie nieprzewidzianych wyników.

Dobre narzędzia

Programista, który zamierza opatrzyć kontrolkę ActiveX podpisem, potrzebuje odpowiednich narzędzi — certyfikatu, ale by go zastosować potrzebny będzie pakiet *ActiveX SDK* (*ActiveX Software Development Kit*) Microsoftu. ActiveX SDK jest zestawem narzędzi koniecznych do podpisywania i testowania plików CAB. Głównymi komponentami tego pakietu są: *makecert.exe*, *cert2spc.exe*, *signcode.exe* oraz *checktrust.exe*. Narzędzia te wchodzą również w skład wprowadzanego dopiero pakietu Microsoft.NET Framework.

- Narzędzie tworzenia certyfikatów (*makecert.exe*) generuje certyfikat X.509, który może służyć jedynie do testowania. Program ten tworzy także publiczny i prywatny klucz na potrzeby podpisów cyfrowych.
- Narzędzie testowe certyfikatów producentów oprogramowania (*Cert2spc.exe*) tworzy certyfikat typu *Software Publisher's Certificate* (*SPC*) z jednego lub większej liczby certyfikatów X.509. Należy pamiętać, że program ten służy jedynie do testów.
- Narzędzie podpisywania plików (*signcode.exe*) służy do podpisywania przenośnego pliku wykonywanego (*portable executable*, *PE*) i nadawania mu żądanych uprawnień, co pozwala twórcom na szczegółową kontrolę ograniczeń nałożonych na ich komponenty. Podpisem można opatrzyć indywidualny komponent lub cały pakiet. Program signcode uruchomiony bez żadnych parametrów włączy kreatora podpisów cyfrowych, który poprowadzi przez cały proces.
- Narzędzie weryfikacji certyfikatów (*chktrust.exe*) sprawdza ważność pliku opatrzonego podpisem *Authenticode*. Jeśli odpowiednie wartości się zgadzają, chktrust uznaje certyfikat za zweryfikowany.

Opisane narzędzia przyspieszą podpisywanie i dystrybucję kontrolek ActiveX.

Bezpieczne kontrolki ActiveX

Komu zaufać, a raczej, kto zaufa mnie? Takie pytanie powinien sobie zadać każdy twórca, który publikuje kontrolki ActiveX. Niezależnie od tego, czy kontrolka znajdzie zastosowanie w internecie czy w korporacyjnej sieci lokalnej, powinna być łatwa w instalacji i informować użytkowników, że jest godna zaufania. Metoda pozwalająca zdobyć to zaufanie nazywa się *podpisywaniem kontrolek*.

Podpisywanie kontrolek

Do podpisania kontrolki konieczny jest cyfrowy certyfikat lub identyfikator (rysunek 9.6) z *Organu certyfikującego*. Dwoma przewodnimi organami certyfikującymi kontrolki ActiveX w USA są VeriSign (*www.verisign.com*) oraz Thawte (*www.thawte.com*). Obydwie firmy oferują różne wersje certyfikatów — w zależności od platformy, na której działa projekt. Na przykład istnieją różne certyfikaty dla technologii Microsoft Authenticode, Netscape Object, Microsoft Office 2000 oraz VBA, Marimba, Macromedia Shockwave, Apple i innych. W chwili obecnej roczny koszt cyfrowego identyfikatora typu Code Signing Digital ID wynosi 400 USD. W przypadku Thawte koszt ich pakietu Developer Certificate wynosi 200 USD, zaś co roku obowiązuje opłata odnawiająca w wysokości 100 USD. W przeciwieństwie do VeriSign, Thawte proponuje teraz uniwersalny certyfikat, służący do podpisywania kodu na wszystkich platformach z wyjątkiem Javy.

Rysunek 9.6. *Ostrzeżenie zawierające identyfikator podpisu kodu*

Firma twierdzi, że pakiet Java 2 Plugin Developer Certificates wciąż nie współpracuje z pozostałymi platformami, w związku z czym dla Javy trzeba wykupić osobny certyfikat. Jeśli jednak w grę wchodzi podpisywanie apletów Javy dla Navigatora (JVM) 1.1.x, wystarczy certyfikat uniwersalny.

Jedną z opcji proponowanych przez VeriSign jest dostęp do serwera stempli czasowych. Ponieważ certyfikat jest ważny jedynie przez rok, to po każdym odnowieniu certyfikatu należy ponownie podpisać kod. Nie jest to jednak prawdą, jeśli podpis w kodzie zawiera stempel czasowy. Oferta darmowego serwera stempli czasowych VeriSign zmniejsza pracę związaną z dbaniem o stary kod. Jest tylko jeden wyjątek — w chwili obecnej Netscape nie obsługuje stempli czasowych, więc kod dla Netscape musi być podpisywany co roku.

Obydwa organy certyfikujące proponują taki sam rodzaj produktu, z których każdy ma zalety, jak Cadillac i Chevrolet — obydwa są dobrymi produktami. Jeden jest tańszy, drugi zawiera więcej udogodnień, ale obydwa pozwolą osiągnąć cel. Europejscy twórcy mogą współpracować z europejskimi organami certyfikującymi. Najpopularniejszymi organami certyfikującymi w Europie są GlobalSign (*www.globalsign.net*) oraz TrustWise (*www.trustwise.com*).

Co zrobić z posiadanym już cyfrowym certyfikatem? Cóż, ponieważ niniejszy rozdział opisuje kontrolki ActiveX, uwaga zostanie skupiona na podpisywaniu kodu dla platformy Microsoftu oraz na Microsoft Authenticode.

Mimo że Thawte i VeriSign funkcjonują jak dwie niezależne firmy i wciąż prowadzą dwie oddzielne linie produktów, Thawte zostało wykupione przez VeriSign, Inc. w grudniu 1999 r.

Jak stosować Microsoft Authenticode

Co to jest i do czego służy Microsoft Authenticode? Authenticode to metoda zdobycia zaufania klientów. Posiadając cyfrowy certyfikat, kod można opatrzyć podpisem. Bez niego użytkownik otrzymałby komunikat o błędzie informujący, że niemożliwe jest ustalenie

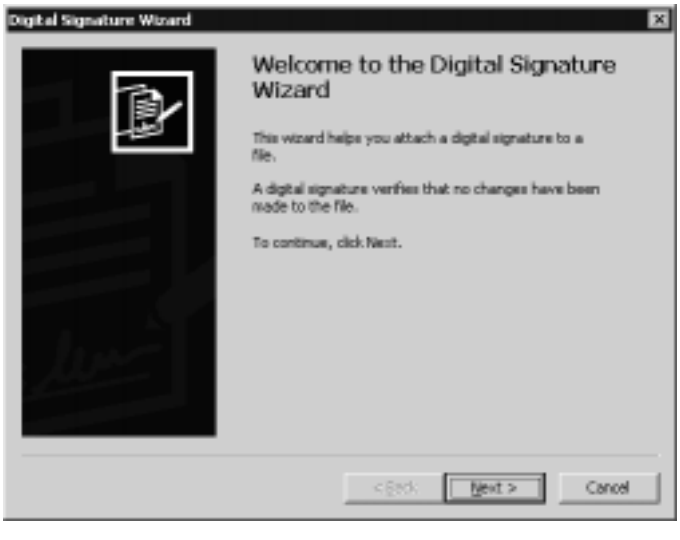
twórcy oprogramowania (rysunek 9.7). Gdy kod posiada podpis, wyświetlane są: informacje o kontrolce, tożsamość i kontakt do twórcy, organ certyfikujący oraz opcjonalnie czas i data podpisania kontrolki. Gwarantuje to użytkownikowi, że kontrolka została udostępniona przez zaufanego producenta oraz że nie była modyfikowana od momentu publikacji.

Rysunek 9.7. *Ostrzeżenie Authenticode*

Jak stosować Microsoft Authenticode? Implementacja technologii Authenticode polega na podpisaniu kodu. Sam proces podpisywania jest bardzo prosty. Ukończona (i w miarę potrzeby umieszczona w pliku *CAB*) kontrolka jest gotowa do podpisu, do czego konieczne będzie narzędzie signcode Microsoftu. Należy wykonać następujące czynności:

1. Po uruchomieniu pliku *signcode.exe*, pojawia się *Kreator podpisów cyfrowych* (rysunek 9.8).

Rysunek 9.8. *Kreator podpisów cyfrowych*



2. Należy wybrać podpisywany plik. Może to być dowolny plik wykonywany (*.exe*, *.ocx* lub *.dll*). Istnieje również możliwość podpisania plików *CAB*, plików katalogów *(CAT)* oraz plików *CTL*.

3. Po wybraniu pliku można zdecydować pomiędzy opcją *Typical* (typowa) oraz *Custom* (niestandardowa). W przypadku korzystania z cyfrowego identyfikatora (otrzymanego od organu autoryzującego) wybiera się opcję *Custom*.
4. Następnie należy wybrać plik certyfikatu (*.cer*, *.crt* lub *.cps*).
5. W kolejnym kroku trzeba podać plik zawierający klucz prywatny *(PVK)*. W tym momencie użytkownik jest proszony o wpisanie hasła. Jeśli hasło zaginęło, trzeba wystąpić o wydanie nowego certyfikatu. Ponieważ jest to dość popularny problem, ponowne wystawienie certyfikatu przez obydwa organy autoryzujące jest darmowe.
6. Należy wybrać algorytm mieszający, który zostanie użyty do stworzenia podpisu. Użytkownik będzie miał w miarę potrzeby możliwość dołączenia dodatkowych certyfikatów.
7. Kolejnym krokiem jest opis danych (bardzo istotne). Są to informacje opisujące kontrolkę, które będą wyświetlane w momencie, gdy użytkownik instaluje kontrolkę.
8. Następnie pora na stempel czasowy konieczny do utrzymania kontrolki w stanie aktywnym po wygaśnięciu certyfikatu. Posiadacze certyfikatu VeriSign mogą korzystać z firmowego serwera stempli czasowych, w przeciwnym razie stempel będzie musiał pochodzić z innego źródła.
9. Na koniec użytkownik może zobaczyć wszystkie ustawienia. Teraz wystarczy kliknąć przycisk *Zakończ*, ponownie wprowadzić hasło certyfikatu i to wszystko — kontrolka ActiveX została podpisana!

Dla tych, którzy nie przepadają za kreatorami, signcode.exe może być również uruchomiony z wiersza poleceń. Wystarczy jedynie podać odpowiednie parametry, a efekty będą takie same. Żeby nie wyprzedzać, omówimy jeszcze jedno zagadnienie. Przed podpisaniem kodu trzeba wiedzieć, jak oznakować kontrolkę.

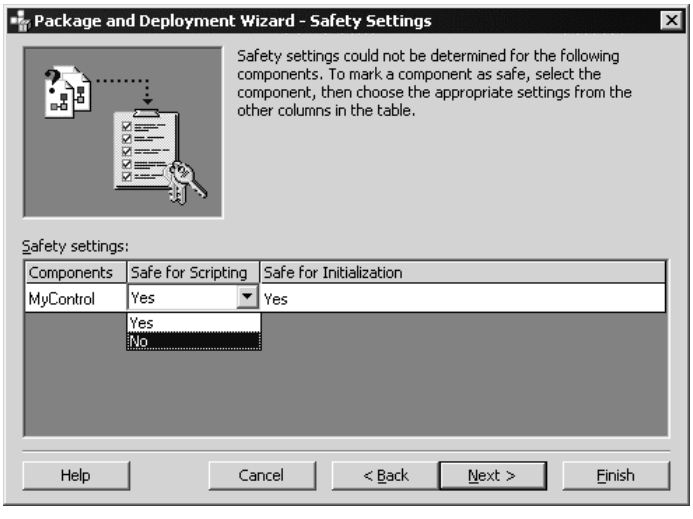
Znakowanie kontrolek

Istnieją dwie metody pozwalające na oznakowanie kontrolki jako bezpiecznej: ustawienia bezpieczeństwa kreatora pakietów i wersji dystrybucyjnych (*Package and Deployment Wizard*) w środowisku programistycznym lub implementacja interfejsu *IObjectSafety*. Najpierw zostanie omówiona łatwiejsza metoda.

Ustawienia bezpieczeństwa

Do tworzenia pakietów w postaci plików *CAB* wystarczy kreator wersji dystrybucyjnych (*Package and Deployment Wizard*). Aby oznaczyć kontrolkę ActiveX jako „bezpieczną ze względu na skrypty" czy „bezpieczną ze względu na inicjalizację", należy wybrać opcję *Yes* w rozwijanej liście obok nazwy kontrolki na ekranie *Safety Settings* kreatora (rysunek 9.9). Oznakowanie kontrolki jako bezpiecznej pozwoli użytkownikom upewnić się, że kontrolka nie wykona w systemie operacji niebezpiecznych. Po wybraniu odpowiednich parametrów bezpieczeństwa klika się przycisk *Dalej*, a kreator zadba o resztę. Plik *CAB* będzie gotowy do instalacji, zaś kontrolka zostanie oznakowana zgodnie z wybranymi ustawieniami.

Rysunek 9.9. *Ekran ustawień bezpieczeństwa kreatora wersji dystrybucyjnych*



IObjectSafety

Drugą metodą oznakowania kontrolki jako bezpiecznej jest implementacja w kontrolce metody IObjectSafety. Jest ona interfejsem komponentu dostępnym z poziomu Microsoft Internet Explorera 4.0 i późniejszych. Zapewnia metody służące do pobierania i ustawiania opcji bezpieczeństwa aplikacji dla systemu Windows. Jest to bardzo prosty interfejs, który posiada jedynie dwie metody:

- `GetInterfaceSafetyOptions`
- `SetInterfaceSafetyOptions`

Nazwy nie pozostawiają wątpliwości co do działania tych metod. `GetInterfaceSafetyOptions` pobiera opcje bezpieczeństwa obsługiwane przez obiekt, jak również bieżące ustawienia opcji bezpieczeństwa obiektu. `SetInterfaceSafetyOptions` pozwala oznakować obiekt jako bezpieczny ze względu na inicjalizację i na skrypty.

W językach Visual Basic 5 i późniejszych najlepszym sposobem jest zastosowanie polecenia `Implements` (listing 9.1). Interfejs IObjectSafety pozwala kontrolce na zwrócenie aplikacji wywołującej metodę (zwanej także obiektem kontenerowym — *Container Object*) odpowiedzi, czy jest bezpieczna, czy nie. Główną zaletą interfejsu IObjectSafety jest możliwość implementacji jednej wersji kontrolki, która zachowuje się bezpiecznie w pewnych okolicznościach i niebezpiecznie w innych. Kontrolka może programowo zmieniać tryby bezpieczeństwa, tak by dostosowywać się do sytuacji. W przeciwieństwie do innych metod oznakowywania kontrolek, sposób ten nie zależy od wpisów rejestru systemowego. Z punktu widzenia bezpieczeństwa, najistotniejszym powodem stosowania interfejsu IObjectSafety jest fakt, że nikt nie może „odpakować" kontrolki i oznaczyć jej jako bezpieczną, podczas gdy taką **nie** jest.

Listing 9.1. *Implementacja interfejsu IObjectSafety w języku Visual Basic*

```
Option Explicit
Implements IObjectSafety
```

```
' IObjectSafety_GetInterfaceSafetyOptions --------------------
Private Sub IObjectSafety_GetInterfaceSafetyOptions(ByVal riid As Long, _
      pdwSupportedOptions As Long, pdwEnabledOptions As Long)
    Dim Rc          As Long
    Dim rClsId      As uGUID
    Dim IID         As String
    Dim bIID()      As Byte

    pdwSupportedOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER Or _
        INTERFACESAFE_FOR_UNTRUSTED_DATA

' Ustaw i zwroc obslugiwane przez obiekt opcje bezpieczenstwa.

    If (riid <> 0) Then
    ' Sprawdz wskaznik do identyfikatora interfejsu.

        CopyMemory rClsId, ByVal riid, Len(rClsId)
        ' Zapisz guid interfejsu do struktury.

        bIID = String$(MAX_GUIDLEN, 0)
        ' Utworz tablice bajtow

        Rc = StringFromGUID2(rClsId, VarPtr(bIID(0)), MAX_GUIDLEN)
        ' Pobierz clsid ze struktury guid.

        Rc = InStr(1, bIID, vbNullChar) - 1
        ' Sprawdz, czy na koncu tekstu nie ma pustych znakow.

        IID = Left$(UCase(bIID), Rc)
        ' Obetnij puste znaki i zamien na wielkie litery dla porownania.

        Select Case IID
        Case IID_IDispatch ' Zadane opcje bezpieczenstwa
            pdwEnabledOptions = IIf(m_fSafeForScripting, _
                INTERFACESAFE_FOR_UNTRUSTED_CALLER, 0)
            Exit Sub
        Case IID_IPersistStorage, IID_IPersistStream, ID IPersistPropertyBag
            pdwEnabledOptions = IIf(m_fSafeForInitializing, _
                INTERFACESAFE_FOR_UNTRUSTED_DATA, 0)
            Exit Sub
        Case Else
            Err.Raise E_NOINTERFACE ' BLAD - opcja nieobslugiwana
            Exit Sub
        End Select
    End If
End Sub
'------------------------------------------------------------

' IObjectSafety_SetInterfaceSafetyOptions -------------------
Private Sub IObjectSafety_SetInterfaceSafetyOptions(ByVal riid As Long, ByVal _
      dwOptionsSetMask As Long, ByVal dwEnabledOptions As Long)
    Dim Rc          As Long
    Dim rClsZd      As uGUID
    Dim IID         As String
    Dim bIID()      As Byte

    If (riid <> 0) Then
    ' Sprawdz wskaznik do identyfikatora interfejsu.
```

```
        CopyMemory rClsId, ByVal riid, Len(rClsId)
        ' Zapisz guid interfejsu do struktury.

        bIID = String$(MAX_GUIDLEN, 0)
        ' Utworz tablice bajtow.

        Rc = StringFromGUID2(rClsId, VarPtr(bIID(0)), MAX_GUIDLEN)
        ' Pobierz clsid ze struktury guid.

        Rc = ZnStr(1, bIID, vbNullChar) - 1
        ' Sprawdz, czy na koncu tekstu nie ma pustych znakow.

        IID = Left$(UCase(bIID), Rc)
        ' Obetnij puste znaki i zamien na wielkie litery dla porownania.

        Select Case IID
        Case IID_IDispatch
            If ((dwEnabledOptions And dwOptionsSetMask) <> INTERFACESAFE FOR UNTRUSTED _
                  CALLER) Then
                Err.Raise E_FAIL ' blad: nieobslugiwane.
                Exit Sub
            End If

            If Not m_fSafeForScripting Then Err.Raise E_FAIL
            ' Czy ten obiekt jest bezpieczny ze wzgledu na skrypty?
            Exit Sub
        Case IID_IPersistStorage, IID_IPersistStream, IID_IPersistPropertyBag
            If ((dwEnabledOptions And dwOptionsSetMask) <> _
                  INTERFACESAFE_FOR_UNTRUSTED_DATA) Then
                Err.Raise E_FAIL ' blad: nieobslugiwane.
                Exit Sub
            End If
            If Not m_fSafeForInitializing Then Err.Raise E_FAIL
            ' Czy ten obiekt jest bezpieczny ze wzgledu na inicjalizacje?
            Exit Sub
        Case Else
            ' Zadany interfejs jest nieznany.
            Err.Raise E_NOINTERFACE ' blad: nieobslugiwane.
            Exit Sub
        End Select
    End If
End Sub
'-------------------------------------------

' FunctionSafeToScript -----------------------
Public Function FunctionSafeToScript() As Boolean
    FunctionSafeToScript = True
End Function
'-------------------------------------------

' FunctionNOTSafeToScript --------------------
Public Function FunctionNOTSafeToScript() As Boolean
    FunctionNOTSafeToScript = True
End Function
'-------------------------------------------
```

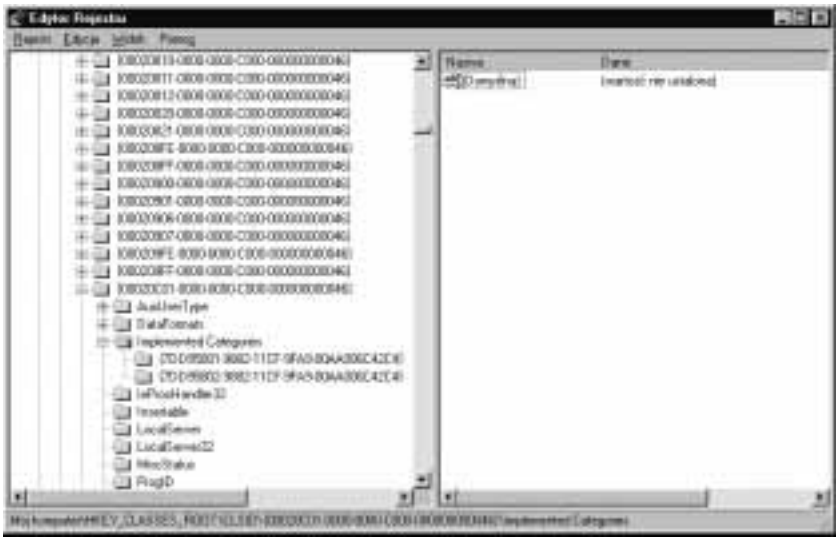
Oznaczanie kontrolek w rejestrze systemowym

Ostatnią opisywaną metodą oznaczania kontrolki jako bezpiecznej jest zastosowanie rejestru systemowego. Mimo że na początku niniejszego rozdziału napisano, iż istnieją tylko dwa sposoby oznakowania kontrolki, niniejsza — trzecia jest tylko rozszerzeniem pierwszej z opisanych metod.

Oznakowanie kontrolki przez kreatora wersji dystrybucyjnych w istocie polega na modyfikacji wpisów w rejestrze systemowym. Należy zrozumieć, że pociąga to za sobą pewne koszty. Przede wszystkim kontrolka, która jest oznakowana w ten sposób, przy każdej inicjalizacji musi odwoływać się do *Rejestru*. To zajmuje czas, zaś gdy w grę wchodzi wyświetlanie zawartości strony WWW, szybkość jest istotnym czynnikiem. Kolejnym problemem związanym z tego rodzaju oznakowywaniem jest fakt, że nie istnieje rozwiązanie pośrednie. Kontrolka jest albo bezpieczna, albo nie. Nie można napisać kontrolki, której parametry bezpieczeństwa zależą od wpisów w rejestrze i która może uchodzić za bezpieczną i jednocześnie niebezpieczną. Trzeba byłoby dostarczać obie wersje — bezpieczną w każdych warunkach i niebezpieczną.

Dla użytkowników, którzy nie mogą się doczekać uruchomienia programu *regedit.exe* i oglądania wszystkiego w rejestrze systemowym, zostaną zaprezentowane odpowiednie czynności. Jedyne, co trzeba zrobić, to utworzyć następujące klucze w polu *CLSID* kontrolki ActiveX, w sekcji *Implemented Categories* (rysunek 9.10).

Rysunek 9.10. *Edytor rejestru Windows*



- Aby oznaczyć kontrolkę jako *bezpieczną ze względu na skrypty*, należy użyć wartości klucza: `7DD95801-9882-11CF-9FA9-00AA006C42C4`.
- Aby oznaczyć kontrolkę jako *bezpieczną ze względu na inicjalizację*, trzeba użyć wartości klucza: `7DD95802-9882-11CF-9FA9-00AA006C42C4`.

To wszystko, choć pamiętajmy, że kontrolka oznaczona jako bezpieczna nie wymaga pozwolenia użytkownika na uruchomienie. Ważna jest zatem ostrożność.

Podsumowanie

Jak się okazuje, z dystrybucją kontrolek ActiveX wiąże się wiele problemów bezpieczeństwa. Wszystkie wynikają z podejścia firmy Microsoft do tych zagadnień. Nadając kontrolkom ActiveX takie same możliwości i prawo dostępu jak użytkownikowi, Microsoft stworzył potężne narzędzie tworzenia kodu przenośnego. Jednak wraz z możliwościami wzrasta też odpowiedzialność, a większość tej odpowiedzialności ponosi programista. To on musi ocenić możliwości kontrolki już w fazie projektowania. Należy unikać powszechnych pomyłek, takich jak błędne oznaczenie kontrolki jako bezpiecznej czy wprowadzenie wersji z błędami.

Poza projektantami także administratorzy systemowi i użytkownicy powinni się przyczynić do ochrony swoich sieci i komputerów osobistych. Administratorzy powinni korzystać ze wszystkich narzędzi dostarczanych z systemami operacyjnymi i rozważyć zastosowanie jakiegoś rodzaju bariery firewall. Administratorzy i użytkownicy mają również dostęp do opcji bezpieczeństwa wbudowanych w programy Internet Explorer, Outlook oraz Outlook Express. Powinni dbać o ustawienia w swoich programach oraz uaktualniać oprogramowanie na swoich komputerach.

Projektanci powinni zapoznać się z dostępnymi narzędziami, które pozwolą opracowywać bezpieczniejsze kontrolki. Cyfrowe certyfikaty połączone z technologią typu Microsoft Authenticode pozwolą osiągnąć ten cel. Podczas pisania kontrolek należy zdawać sobie sprawę z różnych metod oznaczania ich jako bezpieczne oraz znać kryteria kwalifikujące kontrolkę jako bezpieczną lub nie. Zalecaną metodą implementacji tego zadania jest oczywiście interfejs IObjectSafety, jednak obydwie metody spełniają swoje zadanie. Jeśli kontrolka jest całkowicie bezpieczna i nie jest w stanie wyrządzić żadnej szkody w macierzystym systemie, powinny wystarczyć ustawienia rejestru. Jeśli jednak istnieje szansa, by kontrolka mogła być użyta w jakimś niewłaściwym celu, naprawdę warto poświęcić dodatkowy czas na implementację interfejsu IObjectSafety. Trzeba pamiętać, że niezależnie od sposobu potraktowania przez projektanta bezpieczeństwa w technologii ActiveX, to on może stanowić ostatnią linię obrony. Dlatego powinno się być tak dokładnym jak to tylko możliwe i nigdy nie przeceniać potencjału kontrolki.

Skrót rozwiązań

Zagrożenia związane z technologią ActiveX

- Umieszczenie apletu Javy w piaskownicy daje pewność, że aplikacja działa we własnej, chronionej przestrzeni adresowej, odizolowanej od takich rzeczy, jak system plików czy inne aplikacje. Z kolei kontrolki ActiveX mają takie same prawa jak użytkownik, który je uruchamia po instalacji. Microsoft nie jest w stanie zagwarantować, że to autor kontrolki będzie jej używał, tak samo nie ma pewności, że kontrolka zostanie użyta zgodnie z przeznaczeniem — na witrynie lub stronach, dla których powstała. Ponadto nie ma pewności, że właściciel strony czy ktokolwiek inny nie zmodyfikował jej zawartości po umieszczeniu na niej kontrolki.

- Po oznakowaniu kontrolki jako bezpiecznej inne aplikacje i kontrolki mają możliwość uruchamiania jej bez pytania użytkownika o zgodę. Fakt, że programista stworzył kontrolkę z myślą o konkretnym zastosowaniu nie znaczy, że ktoś inny nie znajdzie dla niej innego zastosowania.
- Powszechnym problemem związanym z kontrolkami ActiveX jest udostępnianie wersji, które nie zostały dokładnie przetestowane i zawierają błędy w rodzaju *przepełnienia buforów*. Warto poświęcić dodatkowy czas na dokładne testy, które dadzą pewność, że kod implementuje prawidłową kontrolę ograniczeń wobec wszystkich wartości, których długość może być zmienna.
- Do nakładania restrykcji na kontrolki mogą służyć mechanizmy stref bezpieczeństwa (*Security Zones*) oraz protokoły SSL.
- Zmienna `CodeBaseSearchPath` w rejestrze systemowym określa miejsca, które system bada w poszukiwaniu kontrolek ActiveX.
- Pakiet IEAK może służyć do definiowania i dynamicznego zarządzania kontrolkami ActiveX.

Metodologia tworzenia bezpiecznych kontrolek ActiveX

- Należy dokładnie dokumentować kontrolki. Projektowane kontrolki powinny również oferować minimalną funkcjonalność, konieczną do zrealizowania zadania.
- Jeśli kontrolka wykonuje jakąkolwiek operację spośród wymienionych poniżej, **nie** powinna być oznaczona jako bezpieczna:
  - Dostęp do informacji o lokalnym komputerze lub użytkowniku.
  - Udostępnianie prywatnych informacji w komputerze lokalnym bądź w sieci.
  - Modyfikacja lub usuwanie informacji z komputera lokalnego bądź sieci.
  - Błędy kontrolki potencjalnie prowadzące do zakończenia działania przeglądarki.
  - Nadmierne zużycie czasu i zasobów, takich jak pamięć.
  - Wykonywanie potencjalnie destrukcyjnych wywołań systemowych (włącznie z uruchamianiem plików).
  - Podstępne wykorzystanie kontrolki i powodowanie nieprzewidzianych efektów.
- Pakiet ActiveX SDK Microsoftu jest zestawem narzędzi, które są konieczne do podpisywania i testowania plików CAB. Jego głównymi komponentami są programy: *makecert.exe*, *cert2spc.exe*, *signcode.exe* oraz *checktrust.exe*. Narzędzia te wchodzą również w skład wchodzącego na rynek pakietu Microsoft.NET Framework.

Bezpieczne kontrolki ActiveX

- Do podpisania kontrolki jest konieczny cyfrowy certyfikat lub identyfikator wystawiony przez organ autoryzujący. Dwoma najważniejszymi amerykańskimi organami autoryzującymi kontrolki ActiveX są VeriSign (*www.verisign.com*) oraz Thawte (*www.thawte.com*).

- Oferta darmowych usług związanych ze stemplami czasowymi firmy VeriSign pozwoli zaoszczędzić trochę pracy przy dbaniu o starszy kod. VeriSign pozwala korzystać ze swojego serwera stempli czasowych także klientom firmy Thawte.
- Istnieją dwie metody pozwalające na oznakowanie kontrolki jako bezpiecznej: ustawienia bezpieczeństwa kreatora pakietów i wersji dystrybucyjnych (*Package and Deployment Wizard*) w środowisku programistycznym lub implementacja interfejsu IObjectSafety.
- Główną zaletą interfejsu IObjectSafety jest możliwość implementacji jednej wersji kontrolki, która zachowuje się bezpiecznie w pewnych okolicznościach i niebezpiecznie w innych. Kontrolka może programowo zmieniać tryby bezpieczeństwa, tak by dostosowywać się do sytuacji. W przeciwieństwie do innych metod oznakowywania kontrolek, sposób ten nie zależy od wpisów rejestru systemowego.

Pytania i odpowiedzi

Odpowiedzi na poniższe pytania, których udzielili autorzy niniejszej książki, mają za zadanie sprawdzenie stopnia opanowania zagadnień przedstawionych w niniejszym rozdziale oraz pomóc w zastosowaniu tych koncepcji w codziennej praktyce.

P: Czy, chcąc podpisać kontrolkę, muszę dokonać zakupu certyfikatu cyfrowego?

O: Jeśli firma planuje udostępniać kontrolkę na zewnątrz, trzeba będzie zakupić od wybranego organu certyfikującego ważny certyfikat. Jednak podczas testów, można skorzystać z narzędzi `Makecert.exe` i `Cert2SPC.exe`, które stworzą certyfikat testowy. Narzędzia te wchodzą w skład pakietu ActiveX SDK.

P: Moja firma posiada certyfikat serwera. Czy można go wykorzystać do podpisania kodu?

O: Nie, nie da się podpisać kodu za pomocą certyfikatu serwera. Certyfikat serwera spełnia inną funkcję niż certyfikat służący do podpisywania kodu. Certyfikat serwera pozwala jedynie na bezpieczną transmisję danych pomiędzy serwerem a klientem. Certyfikat do podpisywania kodu lub identyfikator potwierdza, że oprogramowanie nie zostało zmodyfikowane od momentu sporządzenia podpisu.

P: Czy istotna jest stosowana wersja Internet Explorera?

O: Tak. Microsoft, wraz z kolejnymi wersjami IE (począwszy od wersji 3.0) publikował różne wersje narzędzi związanych z podpisami kontrolek. Należy się upewnić, że wykorzystywana wersja przeglądarki jest właściwa.

P: Czy mój nowy certyfikat może służyć do podpisywania kontrolek na potrzeby IE 3.x?

O: Nie. Wersja mechanizmu Authenticode wchodząca w skład przeglądarek IE 3.x nie obsługuje nowych certyfikatów.

P: Jakie są zalety opatrywania kontrolek stemplem czasowym?

O: Stempel czasowy daje pewność, że kod będzie ważny po zakończeniu okresu ważności certyfikatu. W przypadku stosowania certyfikatu VeriSign usługa stempli czasowych jest bezpłatna i zapewnia, że w przypadku ostemplowania kontrolki stemplem czasowym nie trzeba martwić się o ponowne podpisywanie kodu w momencie, gdy okres ważności certyfikatu dobiega końca. Dzięki stemplom czasowym użytkownik będzie w stanie stwierdzić różnicę pomiędzy kodem podpisanym za pomocą przeterminowanego certyfikatu, a kodem podpisanym certyfikatem, który był ważny w momencie sporządzania podpisu.

P: Posiadam certyfikat Thawte i chcę opatrzyć kod stemplem czasowym. Czy Thawte zapewnia stemple czasowe?

O: Nie, jednak VeriSign pozwala klientom Thawte na korzystanie ze swojego serwera stempli czasowych. Aby dostać się do serwera, należy w wierszu poleceń lub w polu serwera stempli czasowych kreatora podpisów cyfrowych wprowadzić po opcji *-t* adres *http://timestamp.verisign.com/scripts/timstamp.dll*.

P: Czy mogę skorzystać z Authenticode dla apletów Javy?

O: Tak, wystarczy aplety Javy umieścić w plikach *CAB*, a odwoływać się do nich z poziomu HTML-a za pomocą znacznika `CABBASE` (zamiast znacznika `ARCHIVE` stosowanego w przeglądarkach Netscape).

P: W jaki sposób przetestować podpis po jego sporządzeniu?

O: Korzysta się z narzędzia chktrust.exe wchodzącego w skład pakietu Microsoft ActiveX SDK. Pozwoli to zweryfikować ważność podpisu przed rozpoczęciem dystrybucji kontrolki.