LESZEK KĘPA
PAWEŁ TOMASIK
SEBASTIAN DOBRZYŃSKI

BEZPIECZEŃSTWO SYSTEMU e-commerce

CZYLI JAK **BEZ RYZYKA** PROWADZIĆ BIZNES W INTERNECIE

one press
Helion

Autorzy: Leszek Kępa, Paweł Tomasik, Sebastian Dobrzyński
Redaktor prowadzący: Barbara Gancarz-Wójcicka
Projekt okładki: Jan Paluch

Fotografia na okładce została wykorzystana za zgodą Shutterstock.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *onepress@onepress.pl*
WWW: *http://onepress.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://onepress.pl/user/opinie/bezsec
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-3873-4

Printed in Poland.

SPIS TREŚCI

BEZPIECZEŃSTWO OGÓLNE

Wspomnieliśmy już wcześniej, że **zagadnienie bezpieczeństwa systemu e-commerce jest złożone**. Składa się na nie:

- bezpieczeństwo fizyczne serwerów, na których znajduje się aplikacja, serwer WWW, baza danych itd.;
- zabezpieczenie systemów operacyjnych, w których są zainstalowane aplikacja, serwer WWW i baza danych;
- takie bezpieczeństwo transmisji, aby danych w trakcie przesyłania nikt nie podsłuchał ani nie zmienił;
- bezpieczeństwo serwera DNS i domeny — to, żeby nikt nie ukradł „nazwy internetowej" serwisu ani nie przekierował ruchu do innego, podrobionego systemu;
- bezpieczeństwo aplikacji, a więc takie jej utworzenie, aby nie można było nią manipulować;
- ciągłość działania, tj. takie zorganizowanie systemu i uzyskanie takiej jego odporności, aby nawet mimo ataku mógł on dalej funkcjonować lub wznowić aktywność;
- bezpieczeństwo organizacyjne, czyli procesy związane z administrowaniem aplikacją, zarządzanie zmianami i wszystko to, co jest dookoła systemu, a ma na niego (znaczący) wpływ;
- bezpieczeństwo prawne, a więc zarządzanie stanem zgodności z prawem.

Tutaj chcemy przedstawić najogólniejsze zagadnienia związane z bezpieczeństwem systemu e-commerce.

INFORMACJE O SYSTEMIE

Zapewne nieraz się zastanawiałeś, jak to się dzieje, że haker w końcu „dopada" określoną witrynę. Mogą być tego dwa główne powody. Pierwszy może być taki, że witryna jest słabo zabezpieczona i stanowi dla hakera łatwy łup — będzie on mógł jej zhakowanie wpisać do ewidencji swoich sukcesów. Drugi to celowy atak na właśnie ten, a nie inny serwis. Na pewno jednak nieodłącznym elementem każdego ataku jest rekonesans, czyli rozpoznanie. Odbywa się to zupełnie tak jak w „realu" — zanim złodziej przystąpi do działania, obserwuje i zbiera informacje. **Każda informacja może być dla hakera przydatna**, a szczególnie o:

- systemie operacyjnym (rodzaj, wersja, uruchomione programy i usługi);
- bazie danych (rodzaj i wersja);
- kodzie aplikacji;
- chronionych częściach aplikacji;
- konfiguracji i zabezpieczeniach systemu.

Znajomość wersji oprogramowania pozwala hakerowi znaleźć luki w zabezpieczeniach — przy odrobinie szczęścia może on trafić na wersję, w której nie zostały one załatane. Do wyszukania luk zabezpieczeń w określonej wersji oprogramowania haker może użyć np. bazy *CVE*[1]. Ty z niej korzystasz, aby wiedzieć, co jest dziurawe, i żeby to zaktualizować, a cyberprzestępcy używają jej, aby mieć informacje o tym, jakie systemy (i jakie ich wersje) „chorują", są podatne na atak.

[1] *https://cve.mitre.org/*

Wiedza o rodzaju bazy danych pozwala z kolei dostosować ataki polegające na wstrzykiwaniu kodu SQL — już sama ta informacja jest istotna, bo niektóre ataki są specyficzne dla rodzaju bazy danych, np. działają w przypadku PostgreSQL, ale nie z MySQL. Nawet rodzaj języka, w którym napisano aplikację, ma znaczenie.

Sprawdźmy, jak może wyglądać atak cyberprzestępcy, który dla zabawy chce spróbować swoich sił w hackingu. Pierwsze, co może on zrobić, to wpisanie w wyszukiwarce ciągu `sql dorks`. Oto przykładowe ciągi tego typu:

```
allinurl:showimg.php?id=
allinurl:view.php?id=
allinurl:website.php?id=
allinurl:hosting_info.php?id=
allinurl:gallery.php?id=
```

Haker kopiuje i wkleja jeden z nich do wyszukiwarki, a następnie otwiera strony będące wynikami wyszukiwania i przeprowadza rekonesans. Najłatwiej jest zacząć od sprawdzenia, czy dany ciąg (określany mianem SQL dorka) zadziała. Przykładowo *http://www.witryna.com/gallery.php?id=1* zawiera SQL dork `gallery.php?id=`. Wystarczy w pasku adresowym na końcu dodać znak apostrofu i w wyniku otrzymujemy:

```
Fatal error: Uncaught exception 'Exception' with message 'SQL Query
failed: SELECT image,title,location,description,height,width from
tblSCAImages WHERE id=83\\\'You have an error in your SQL syntax; check
the manual that corresponds to your MySQL server version for the right
syntax to use near '\\\'' at line 1' in
/home/sca/wwwlib/gallery_lib.php:16 Stack trace: #0
/home/sca/www/gallery.php(21): sca_gallery_get_image_meta('83\'') #1
{main} thrown in /home/sca/wwwlib/gallery_lib.php on line 16
```

Mamy tu informacje o błędach i już coś wiemy o witrynie — wiemy, że jest to MySQL, a także o nazwach tabel i ich kolumn, o funkcjach PHP, a nawet o systemie plików. To się przyda. Sprawdźmy, jak witryna internetowa zareaguje na wpisanie błędnego

adresu strony *http://www.witryna.com/coscokolwiek*. Tu też otrzymujemy komunikat:

```
Not Found

The requested URL /coscokolwiek was not found on this server.
Apache/2.2.3 (CentOS) Server at www.allspeedperformance.com Port 80
```

Znamy już wersję serwera Apache. Zobaczmy jeszcze, jakie części witryny są chronione. Wpiszmy *http://www.witryna.com/robots.txt*. Otrzymamy:

```
User-agent: *
Disallow: /checkout/
```

To tylko przykład tego, jak można zacząć. Dalej można „grzebać" w cookies, użyć programu `sqlmap`, ręcznie wstrzykiwać kod... Możliwości jest wiele. To pokazuje, że ujawnianie zbyt wielu informacji może działać na szkodę systemu, bo pozwala przeprowadzić rekonesans. Dla hakera może być przydatna w zasadzie każda informacja, zadbaj więc o to, aby nie wypuszczać na zewnątrz komunikatów o błędach wraz z ich detalami oraz informacjami o wersji oprogramowania. Pozwala to hakerowi odnaleźć słabe punkty danej wersji.

Jak wprowadzić hakera w błąd

Uważamy, że skoro podstawą ataków jest rekonesans, to może warto atakującego wyprowadzić w pole. Gdy widać wyraźnie, że coś jest porządnie chronione, to pewnie jest warte tego, aby to ukraść. Można to wykorzystać do stworzenia pułapki na cyberprzestępcę. Będzie on długo łamał zabezpieczenia, aby w efekcie uzyskać dostęp do nic nie wartych informacji lub uruchomić alarm informujący, że dzieje się coś złego. Takie rozwiązanie nazywa się z angielskiego *honeypot*. Taką pułapkę można przykładowo zastawić, korzystając z pliku *robots.txt*, który służy do zezwalania na indeksowanie lub

do wykluczania z indeksowania tych części witryny, na których przeszukiwanie nie chcemy pozwolić tzw. robotom (np. botowi indeksującemu strony dla Google). Plik ten znajduje się w roocie (katalogu głównym) witryny, np. *http://ecommerce/robots.txt*. Można go użyć do zmylenia przeciwnika, wpisując w nim fałszywe odniesienia, a następnie monitorować te fałszywie podstawione punkty.

```
User-agent: *
Crawl-delay: 10
Disallow: /strona_administracyjna/
Disallow: /admin_password.txt
```

Inną możliwością „wkręcania" atakującego jest wprowadzanie go w błąd co do wykorzystanego oprogramowania. Przykładowo po zmianie rozszerzenia i nagłówków kod PHP może udawać inny kod:

```
<?php
error_reporting(0);
header("X-Powered-By: ASP.NET");
?>
```

Należy wtedy jeszcze pamiętać, aby korzystając z `session_name()`, ustawić nazwę sesji na przykład na `SessionID`, ponieważ domyślne `PHPSESSID` od razu wskazuje na PHP.

Rewrite

Parametry kategorii, subkategorii i rozmaitych podstron występujące w adresie URL można z łatwością ukryć, korzystając z `mod_rewrite`[2]. Jeśli Twój system prezentuje adres w postaci *http://strona/news.php?kategoria=10&rodzaj=4*, to można to zmienić, zapisując przykładowo w *.htaccess*[3] na serwerze następujący ciąg:

---

[2] Coraz częściej pojawiają się opinie, że moduł ten umiarkowanie wpływa na bezpieczeństwo, dlatego trzeba mieć zainstalowaną aktualną wersję i stosować go ostrożnie.

[3] Nazwa pliku .htaccess zaczyna się od kropki. Obecność kropki na początku nazwy pliku (lub folderu) oznacza, że jest to plik (folder) ukryty.

```
RewriteEngine On
RewriteRule ^([a-z0-9-_]+),([a-z0-9-_]+),([a-z0-9-_]+).html$
$1.php?kategoria=$2&rodzaj=$3 [L,NC,NS]
```

Zmienne `$` z cyfrą odpowiadają poszczególnym nawiasom reguł-ki, natomiast `^` rozpoczyna, a `$` kończy wyrażenie regularne. Jeszcze ciekawszy wydaje się zapis

```
RewriteRule ^artykul/([0-9]+)_(.*)\.html$ articles.php?id=$1
```

W tym przykładzie `/artykul/24_wielka_wojna` i `/artykul/24_protesty_acta` zostaną zamienione na `/articles.php?id=24` (odniesienie `$2` jest ignorowane). Jak widzisz, jeśli dobrze wykorzystasz ten moduł, to ciągi sql-dork nie będą w serwisie widoczne.

ZAPASOWE KOPIE DANYCH

Znamy osoby, które pisząc pracę magisterską, nie wykonały zapasowej kopii danych. Po stracie plików (półrocznej pracy) musiały napisać ją w ciągu miesiąca. W takich przypadkach zwykliśmy żartować, że **ludzie dzielą się na tych, którzy robią kopie zapasowe, i na tych, którzy będą je robić**. Bezpowrotna utrata danych jest jedną z najgorszych rzeczy, jakie się mogą przedsiębiorcy przydarzyć — może to prowadzić nawet do bankructwa. Skradzione komputery można kupić, a danych niestety nie. W tej materii świadomość nie jest jednak zbyt wysoka, a sam temat kopii zapasowych (zwanych backupami) staje się ważny dopiero wtedy, gdy coś się wydarzy. Tak samo jest w przypadku systemów e-commerce, szczególnie tych małych i średnich. Tymczasem ciągły i poprawny proces wykonywania kopii zapasowych oraz testowania, czy da się z nich odtworzyć dane konieczne do prowadzenia biznesu, pozwala zminimalizować ryzyko ich bezpowrotnej utraty. Wiedzą to bez wątpienia ci, którzy interesują się zagadnieniami zachowania ciągłości działania (tzw. BCM[4]).

[4] BCM — ang. *business continuity management*.

W jakiej sytuacji kopia danych może się przydać? Chyba najlepiej odpowiedzieć, że w każdej:

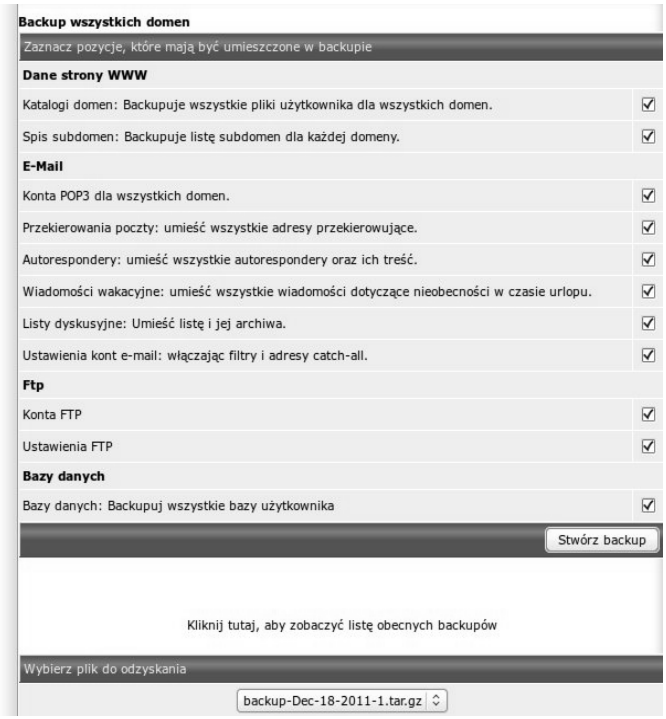
- awaria urządzeń (najczęściej ulegają jej nośniki danych — dyski twarde);
- błąd ludzki, czyli utrata danych w wyniku niecelowego działania pracownika (coś się niechcący usunęło);
- sabotaż — utrata danych w wyniku celowego szkodliwego działania na przykład niezadowolonego pracownika;
- działanie wirusów (oprogramowania złośliwego), których celem może być usunięcie bądź zaszyfrowanie danych, aby uniemożliwić do nich dostęp;
- atak hakerów, czyli celowe usunięcie, zmodyfikowanie czy zniszczenie danych.
- kradzież sprzętu i utrata nośników danych;
- zdarzenie losowe — uszkodzenie sprzętu w wyniku przepięcia, zalania itp.

Jedno z praw Murphy'ego mówi, że *jeśli coś może pójść źle, to z pewnością pójdzie źle*, więc nie ma co liczyć, że nic się nie zdarzy. Na pewno się zdarzy — to tylko kwestia czasu.

Co należy backupować?

Przede wszystkim wykonuje się kopie zapasowe danych biznesowych. Będą to bazy danych, pliki i katalogi serwisu e-commerce. W przypadku hostingu zajmuje się tym hostingodawca, ale jeśli serwery są w całości pod Twoim władaniem, to zadanie to spada na Ciebie (rysunek 2.1).

Jeśli zarządzasz całym systemem, niezbędny będzie backup konfiguracji aplikacji, bazy danych, systemów operacyjnych czy też urządzeń sieciowych.



Rysunek 2.1. Hosting — możliwość wykonania samodzielnie backupu z poziomu panelu hostingowego (DirectAdmin)

To, jak często należy wykonywać zapasową kopię danych biznesowych, zależy od systemu i jego specyfiki. Trzeba odpowiedzieć sobie na pytanie o to, z jakiego okresu dane mogą zostać utracone. Jeśli serwis jest mało aktywny, to zapasowa kopia danych wykonywana raz na tydzień też może być dobra, jednak w przypadku aktywnych systemów na pewno powinna ona być tworzona codziennie. Pamiętaj, że **serwisy pracujące w klastrze to nie backup**; nie stanowi go też np. disk mirroring.

Dane dotyczące konfiguracji można backupować nieco rzadziej, ale należy to robić przed każdą zmianą i po niej.

Jeśli chodzi o hosting, to większość hostingodawców — jeśli nie wszyscy — tworzy zapasowe kopie danych i udostępnia je użytkownikowi (rysunek 2.2).

Rysunek 2.2. Backup danych dostępny w usłudze hostingu

W większości przypadków dostępny jest także backup na żądanie. Przydaje się on na przykład przed dokonaniem w serwisie istotnych zmian, które są na tyle ryzykowne, że lepiej zapewnić sobie kopię. Operacją taką jest choćby aktualizacja systemu czy zmiana struktury bazy danych. System informatyczny to nie człowiek — większości operacji nie powinno się wykonywać na „żywym organizmie".

Kopia zapasowa może być niewiele warta, jeżeli okaże się, że została nieprawidłowo wykonana, dlatego okresowo należy sprawdzać, czy w kopii zapisane są potrzebne dane i czy da się je odtworzyć. Za bardzo ważne uważamy to, aby backup był przechowywany także poza siedzibą firmy, nawet w domu, jeśli zostaną zapewnione odpowiednie warunki. Dzięki temu w przypadku dużej awarii będzie

możliwe szybkie odtworzenie systemu na przykład na innym sprzęcie i (lub) w innej lokalizacji.

Kopia zapasowa danych zawiera wszystko to, co jest w oryginalnym środowisku e-commerce, więc mogłaby być cenną zdobyczą dla konkurencji. W związku z tym backup powinien być chroniony na poziomie nie niższym niż sam system e-commerce. Dostęp do niego powinny mieć tylko wybrane osoby. Dobrze by było, aby dane na nośnikach były szyfrowane, szczególnie jeżeli mają być one przechowywane w domu. Zapewnienie tego jest akurat proste, bo cały backup można zabezpieczyć hasłem:

```
~# gpg -c backup.tar  #zaszyfrowanie
~# gpg backup.tar.gpg #odszyfrowanie
```

Nawiasem mówiąc, polecenie `gpg` domyślnie kompresuje dane przed szyfrowaniem. Wystarczające jest też spakowanie danych do pliku ZIP i zabezpieczenie go hasłem. Dzięki temu jest małe ryzyko, że dane, za pomocą których można by utworzyć „duplikat firmy", wpadną w niepowołane ręce.

Warto też zauważyć, że do zabezpieczania plików można użyć pakietu *openssl*. Przykładem są następujące polecenia do szyfrowania i (później) odszyfrowywania danych:

```
~# openssl aes-128-cbc -salt -in plik.tar -out plik.tar.aes
enter aes-128-cbc encryption password:
Verifying - enter aes-128-cbc encryption password:
~# openssl aes-128-cbc -d -salt -in plik.tar.aes -out plik.tar
```

Można też zrobić to nieco inaczej — „starować" katalog, spakować do pliku ZIP, później zaszyfrować:

```
~# tar -zcf - caly_katalog | openssl aes-128-cbc -salt -out
caly_katalog.tar.gz.aes
```

a w końcu odszyfrować:

```
~# openssl aes-128-cbc -d -salt -in caly_katalog.tar.gz.aes | tar -xz -f
-
```

Nie zalecamy używania opcji `-k "hasło"` po wyrażeniu `aes-128-cbc` — wtedy co prawda unikamy interaktywnego pytania o hasło, ale prowadzi to do zapisania go na przykład w pliku *.history*[5]. Można też użyć `aes-256-cbc` zamiast `aes-128-cbc`, jeśli jest potrzebne silniejsze szyfrowanie, ale naszym zdaniem rzadko kiedy jest to konieczne.

Mówiliśmy o backupie wszystkich danych, ale może się też zdarzyć, że pracując na pojedynczych plikach (np. kodzie PHP), będziesz w trakcie ich modyfikacji tworzył podręczne kopie zapasowe. Chodzi nam o żywy organizm, tj. działający system e-commerce. Przykładowo edytując plik *dbconnect.inc*, możesz zrobić kopię tego pliku i zapisać ją jako *dbconnect.inc.old*. Jest to bardzo niebezpieczne, gdyż może się okazać, że o ile pliki **.inc* serwer może blokować przy użyciu dyrektywy `<Files "*.inc">`, to może tego nie robić dla plików **.old* czy **.backup*. Wtedy atakujący będzie mógł je odczytać przez przeglądarkę, wpisując na przykład adres *http://ecommerce/dbconnect.inc.old*. Powinieneś więc dodać do konfiguracji webserwera odpowiednią dyrektywę i konsekwentnie trzymać się nazewnictwa takich „backupów" tymczasowych (podręcznych).

Backup haseł

Hasło można zapomnieć, więc je też warto w pewien sposób backupować, czyli zapisywać. Najlepiej użyć to tego oprogramowania — nie zalecamy przechowywania haseł w postaci jawnej w notesach, na „żółtych karteczkach" etc. Korzyści związane z ich zapisywaniem z użyciem elektronicznego sejfu są dość duże — możesz tworzyć unikalne, złożone hasła dla każdego serwisu i nie przejmować się tym, że je zapomnisz. Elektroniczny sejf to po prostu zaszyfrowany plik z hasłami; pisaliśmy już wcześniej o programie KeePass. **Musisz mieć jego backup!** Najlepiej mieć ich kilka

[5] W pliku *.history* przechowywana jest historia wszystkich poleceń wydawanych w oknie terminalu.

i w różnych miejscach — wtedy prawdopodobieństwo jednoczesnej utraty ich wszystkich jest niewielkie.

Piszemy o tym, bo przy tworzeniu hasła na przykład do połączenia z bazą systemu e-commerce niekiedy brakuje nam fantazji. KeePass „wymyśli" hasło za Ciebie (rysunek 2.3), a ponieważ jednocześnie je zachowa, nie będziesz musiał przejmować się tym, jakie ono jest, i będziesz mógł pozwolić sobie na to, by było *hardcore'owe* — długie i wręcz niemożliwe do zapamiętania. Zalecamy, aby tworzyć z użyciem tego programu przede wszystkim hasła administracyjne (a najlepiej wszystkie).

Rysunek 2.3. Generowanie haseł przy użyciu programu KeePass

Na koniec warto jeszcze wspomnieć o tym, że plik z hasłami powinien być zabezpieczony porządnym hasłem i przechowywany poza serwerem, na którym znajduje się Twoja aplikacja.

NA STRAŻY TWOJEGO **E-BIZNESU**

Biznes w Internecie to świetna alternatywa dla sprzedaży towarów czy usług w sposób tradycyjny. Kupować w sieci można taniej, szybciej, wygodniej, a rynek odbiorców zwiększa się wielokrotnie w stosunku do tradycyjnego. No i oczywiście „prowadzić e-biznes każdy może". Aż do pierwszego potknięcia, często związanego z niedostatecznym zabezpieczeniem systemu, na podstawie którego pracuje sklep czy platforma e-biznesowa.

Problemy z bezpieczeństwem systemów e-commerce mogą mieć wiele rozmaitych przyczyn.

- Dostawcy mają dostarczyć jak najtaniej w pełni funkcjonalny system i nie są rozliczani za priorytetowe potraktowanie kwestii bezpieczeństwa.
- Ważna jest presja czasowa — im szybciej system będzie dostarczony, tym szybciej zacznie przynosić firmie zyski.
- Materia bezpieczeństwa jest skomplikowana — nie każdy zna się na tych zagadnieniach.
- Mało znane i trudne do przyswojenia bywają zagadnienia prawne — niewielu programistów czy analityków rozumie prawne aspekty prowadzenia przedsiębiorstwa e-commerce.

W książce *Bezpieczeństwo systemu e-commerce* znajdziesz przede wszystkim informacje o mechanizmach prewencyjnych, czyli możliwych sposobach zabezpieczania systemu. Poznasz także wymogi prawne obowiązujące osoby handlujące przez Internet. Autorzy rozpatrują je od strony aplikacji e-commerce, lecz także szerzej (opisują m.in. kwestię zapisów w regulaminie sklepu internetowego).

Bardzo wartościowa publikacja, łącząca kwestie prawne oraz informatyczne, z naciskiem na aspekty praktyczne, pomocna przy projektowaniu rozwiązań na użytek małych i średnich przedsiębiorców.

dr Stefan Szyszko

polski i unijny ekspert w dziedzinie ochrony informacji, ze szczególnym uwzględnieniem ochrony danych osobowych w sektorze ubezpieczeniowym

Informacje zawarte w tym opracowaniu mogą się przydać przedsiębiorcom zamierzającym wejść ze swoją ofertą w świat rozwiązań internetowych lub zmieniającym sposób świadczenia usług. Mogą się też przydać administratorom wdrażającym i eksploatującym systemy e-commerce oraz służyć jako przewodnik osobom zainteresowanym ochroną informacji w systemach e-biznesu.

Maciej Kołodziej

administrator bezpieczeństwa informacji w spółce Nasza Klasa, specjalista informatyki śledczej

Publikacja z pewnością przyczyni się do rozwoju kultury bezpieczeństwa informacyjnego. To bardzo ważna książka dla wszystkich zaangażowanych w biznes online, zarówno tych prowadzących sklepy internetowe, jak i przygotowujących dla nich aplikacje i rozwiązania informatyczne.

Marcin Olszewik

dyrektor zarządzający w Allianz Direct New Europe

książkiklasybusiness

Nr katalogowy: 8535

Księgarnia internetowa:
http://onepress.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

one press

Sprawdź najnowsze promocje:
http://onepress.pl/promocje
Książki najchętniej czytane:
http://onepress.pl/bestsellery
Zamów informacje o nowościach:
http://onepress.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: onepress@onepress.pl
http://onepress.pl

Cena 44,90 zł

ISBN 978-83-246-3873-4

9 788324 638734