BEZPIECZEŃSTWO APLIKACJI INTERNETOWYCH DLA PROGRAMISTÓW

RZECZYWISTE ZAGROŻENIA, PRAKTYCZNA OCHRONA

MALCOLM McDONALD

no starch press

Helion

Tytuł oryginału: Web Security for Developers: Real Threats, Practical Defense

Tłumaczenie: Joanna Zatorska

ISBN: 978-83-283-7803-2

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

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

Printed in Poland.

Spis treści

2

Jak działa internet

JEŚLI CHCESZ ZOSTAĆ EKSPERTEM W DZIEDZINIE BEZPIECZEŃSTWA INTERNETOWEGO, MUSISZ DOBRZE ZROZUMIEĆ TECHNOLOGIE I PROTOKOŁY SIECIOWE, NA KTÓRYCH OPIERA SIĘ INTERNET. W tym rozdziale opisuję zbiór protokołów internetowych, które sterują wymianą danych między komputerami w sieci. Omawiam także połączenia stanowe oraz szyfrowanie, czyli kluczowe aspekty nowoczesnej sieci. Opisując wspomniane komponenty, wskażę, gdzie zwykle pojawiają się luki w zabezpieczeniach.

Zbiór protokołów internetowych

W czasach, gdy internet dopiero raczkował, wymiana danych była zawodna. Pierwsza wiadomość wysłana przez sieć *Advanced Research Projects Agency Network* (ARPANET), poprzednika internetu, była poleceniem LOGIN przeznaczonym dla zdalnego komputera na Uniwersytecie Stanforda. Sieć wysłała dwie pierwsze litery, LO, po czym uległa awarii. Był to niefortunny finał dla armii USA, szukającej sposobu na połączenie zdalnych komputerów, żeby umożliwić wymianę informacji nawet po potencjalnym sowieckim ataku nuklearnym, który doprowadziłby do wyłączenia wielu elementów sieci.

Aby rozwiązać ten problem, inżynierowie sieciowi opracowali protokół *Transmission Control Protocol* (TCP), gwarantujący niezawodną wymianę informacji między komputerami. TCP jest jednym z ok. 20 protokołów sieciowych, które określa się wspólnym mianem *zbioru protokołów internetowych*. Komputer wysyłający wiadomość do innego komputera za pośrednictwem protokołu TCP dzieli ją na pakiety danych, które są wysyłane do adresu docelowego. Komputery tworzące internet przesyłają każdy pakiet w kierunku docelowym bez potrzeby przetworzenia całej wiadomości.

Gdy komputer docelowy odbierze pakiety, składa je w całość na podstawie *numeru sekwencji* z każdego pakietu. Za każdym razem, gdy adresat otrzymuje pakiet, wysyła potwierdzenie. Jeśli nie odeśle potwierdzenia otrzymania pakietu, nadawca wyśle go ponownie, możliwe, że inną drogą w sieci. Dzięki temu protokół TCP umożliwia komputerom przesyłanie danych przez sieć, która jest zawodna z założenia.

W miarę rozwoju internetu protokół TCP doczekał się znaczących ulepszeń. Pakiety są obecnie wysyłane z sumą kontrolną, na bazie której odbiorcy mogą wykrywać naruszenie danych i ocenić, czy należy ponownie przesłać pakiety. Ponadto nadawcy dostosowują tempo wysyłania danych do szybkości ich odbioru. (Serwery internetowe są zwykle o kilka rzędów wielkości szybsze niż komputery klientów odbierających wiadomości, dlatego oprogramowanie serwerów musi uwzględniać możliwości klientów).

UWAGA *TCP nadal jest najpopularniejszym protokołem internetowym ze względu na gwarancje dostarczenia danych, ale obecnie w internecie wykorzystuje się też inne protokoły. Przykładem jest nowszy protokół* User Datagram Protocol *(UDP), który celowo umożliwia porzucanie pakietów, dzięki czemu dane można przesyłać strumieniem o stałej szybkości. Protokół UDP jest zwykle używany do streamingu wideo na żywo, ponieważ klienci wolą pominąć kilka klatek, jeśli dzięki temu unikną opóźnień w transmisji podczas wzmożonego ruchu w sieci.*

Adresy protokołu internetowego

Pakiety danych w internecie są przesyłane do *adresów IP* (*Internet Protocol*), czyli ciągów liczbowych przypisanych do wszystkich komputerów połączonych z internetem. Każdy adres IP musi być unikalny, dlatego nowe adresy IP są przydzielane w sformalizowany sposób.

Na najwyższym poziomie znajduje się organizacja *Internet Corporation for Assigned Names and Numbers* (ICANN), która rezerwuje bloki adresów IP dla organów regionalnych, a te następnie przydzielają bloki adresów dostawcom usług internetowych (ISP — *internet service provider*) oraz firmom hostingowym ze swojego regionu. Gdy łączysz się z internetem poprzez przeglądarkę, dostawca internetu przydziela Twojemu komputerowi adres IP, który nie zmienia się przez kilka miesięcy. (Dostawcy internetu zwykle rotacyjnie przydzielają adresy IP klientom). Także firmy hostujące treści internetowe otrzymują adresy IP dla każdego serwera, którym łączą się z siecią.

Adresy IP są liczbami binarnymi, zwykle zapisywanymi w składni *IPv4* (*IP version 4*), która umożliwia utworzenie 2^{32} (4 294 967 296) adresów. Na przykład serwer nazwy domeny Google ma adres 8.8.8.8. Ponieważ adresy IPv4 są zużywane w nieodnawialnym tempie, w internecie przechodzi się na adresy *IPv6* (*IP version 6*), umożliwiające łączenie większej liczby urządzeń. Adresy te mają postać ośmiu grup cyfr szesnastkowych, rozdzielonych dwukropkami (np. 2001:0db8:0000:0042:0000:8a2e:0370:7334).

System nazw domen

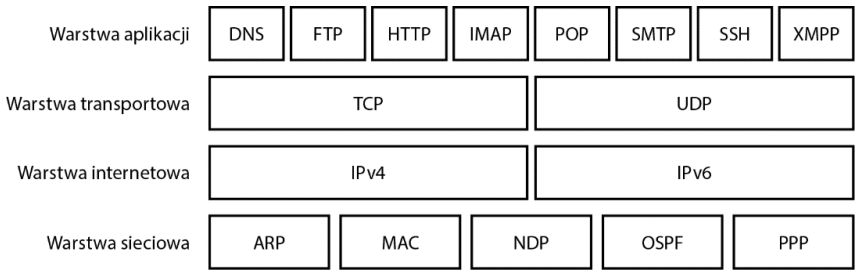
Przeglądarki i inne programy połączone z internetem rozpoznają adresy IP i przekierowują do nich dane, jednak ludziom trudno jest je zapamiętać. Aby ułatwić ludziom rozpoznawanie adresów witryn internetowych, stosuje się globalny katalog zwany systemem nazw domen (*Domain Name System* — DNS), który odwzorowuje domeny czytelne dla ludzi, np. *example.com*, na adresy IP, takie jak 93.184.216.119. Nazwy domen są po prostu zamiennikami adresów IP. Nazwy domen, podobnie jak adresy IP, są unikalne i przed użyciem trzeba je zarejestrować w organizacjach zwanych *rejestratorami domen*.

Gdy przeglądarka po raz pierwszy natknie się na nazwę domeny, sprawdza ją na lokalnym *serwerze nazw domen* (zwykle hostowanym przez ISP), a następnie zapisuje wynik w pamięci podręcznej, aby uniknąć czasochłonnego sprawdzania w przyszłości. Istnienie mechanizmu pamięci podręcznej dla nazw domen wydłuża czas propagacji w internecie nowych domen lub zmian w istniejących. Czas potrzebny na propagację tych zmian jest kontrolowany przez zmienną *TTL* (*time-to-live*), zdefiniowaną w rekordzie DNS. Na podstawie tej zmiennej wiadomo, kiedy pamięć podręczna DNS powinna uznać, że rekord został unieważniony. Mechanizm pamięci podręcznej DNS umożliwia przeprowadzanie ataków zwanych *zatruwaniem DNS*, polegających na celowym naruszaniu lokalnej pamięci podręcznej DNS, aby przekierować dane do serwera kontrolowanego przez atakującego.

Serwery nazw domen nie tylko zwracają adresy IP dla określonych domen, ale również hostują *rekordy nazw kanonicznych* (*CNAME*), które mogą opisywać aliasy domen. Dzięki temu wiele nazw domen może dotyczyć tego samego adresu IP. DNS może też ułatwiać trasowanie wiadomości e-mail za pośrednictwem rekordów *MX* (*mail exchange*). W rozdziale 16. opisuję, jak rekordy DNS mogą ułatwić walkę z niepożądanymi wiadomościami e-mail (spamem).

Protokoły warstwy aplikacji

TCP umożliwia niezawodną wymianę danych w internecie między dwoma komputerami, ale nie narzuca sposobu interpretacji tych danych. W tym celu obydwa komputery muszą wspólnie ustalić mechanizm wymiany informacji poprzez inny protokół wyższego poziomu. Protokoły bazujące na protokole TCP (lub UDP) nazywa się *protokołami warstwy aplikacji*. Rysunek 2.1 przedstawia hierarchię protokołów warstwy aplikacji w odniesieniu do protokołu TCP w zbiorze protokołów internetowych.



Rysunek 2.1. Różne warstwy wchodzące w skład protokołów internetowych

Protokoły niższego poziomu ze zbioru protokołów internetowych definiują podstawowe trasowanie danych w sieci, natomiast protokoły wyższego poziomu, czyli z warstwy aplikacji, zapewniają dalszą strukturyzację dla aplikacji wymieniających dane. Wiele rodzajów aplikacji wykorzystuje TCP jako mechanizm przesyłania danych w internecie. Na przykład wiadomości e-mail są wysyłane za pomocą protokołu Simple Mail Transport Protocol (SMTP), programy obsługujące natychmiastową komunikację wykorzystują Extensible Messaging and Presence Protocol (XMPP), pobieranie danych z serwerów plików odbywa się poprzez File Transfer Protocol (FTP), a serwery WWW wykorzystują HyperText Transfer Protocol (HTTP). Ponieważ koncentrujemy się głównie na technologii webowej, przyjrzyjmy się bliżej protokołowi HTTP.

HyperText Transfer Protocol

Serwery WWW wykorzystują *HyperText Transfer Protocol* (HTTP) do przesyłania stron WWW oraz ich zasobów do *agentów użytkowników*, takich jak przeglądarki internetowe. W trakcie komunikacji poprzez HTTP agent użytkownika wysyła *żądanie* o określone zasoby. Serwery WWW przyjmują te żądania i zwracają *odpowiedzi* zawierające żądane zasoby lub kod błędu, jeśli nie uda się spełnić żądania. Zarówno żądania, jak i odpowiedzi HTTP mają postać normalnych wiadomości tekstowych, chociaż zwykle przed wysłaniem są kompresowane i szyfrowane. Wszystkie exploity opisane w tej książce w pewnym stopniu korzystają z protokołu HTTP, dlatego warto poznać szczegółowy sposób wymiany żądań i odpowiedzi podczas komunikacji z użyciem HTTP.

Żądania HTTP

Żądanie HTTP wysyłane przez przeglądarkę zawiera następujące elementy:

Metoda — określana również mianem *czasownika*, definiuje akcję, którą serwer powinien wykonać po otrzymaniu żądania od agenta użytkownika.

Universal resource locator (URL) — definiuje zasób, który należy przetworzyć lub pobrać.

Nagłówki — zawierają metadane, np. typ treści, jakiego oczekuje agent użytkownika, lub informują, czy agent akceptuje skompresowane odpowiedzi.

Ciało — jest to opcjonalny komponent zawierający dodatkowe dane, które trzeba wysłać na serwer.

Przykładowe żądanie HTTP jest przedstawione na listingu 2.1.

Listing 2.1. Proste żądanie HTTP

```
GET❶ http://example.com/❷
❸ User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
❹ Accept: text/html,application/xhtml+xml,application/xml; */*
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
```

Metoda ❶ i URL ❷ są zdefiniowane w pierwszym wierszu. Następnie w osobnych wierszach znajdują się nagłówki HTTP. Nagłówek User-Agent ❸ informuje witrynę WWW o typie przeglądarki wysyłającej żądanie. Nagłówek Accept ❹ informuje witrynę o typie treści, jakiego oczekuje przeglądarka.

Żądania wykorzystujące metodę GET — zwane skrótowo żądaniami GET — są najpopularniejszymi typami żądań w internecie. Żądania GET pobierają z serwera WWW pewne zasoby, zdefiniowane określonymi adresami URL. Odpowiedź na żądanie GET zawiera zasób. Może to być strona WWW, obraz, a nawet wynik wyszukiwania. Przykładowe żądanie z listingu 2.1 służy do wczytania strony głównej witryny *example.com*. Żądanie to zostałoby wygenerowane, gdyby użytkownik wpisał **`example.com`** na pasku nawigacji przeglądarki.

Jeśli przeglądarka musi wysłać dane na serwer, a nie tylko je pobrać, zwykle wykorzystuje żądanie POST. Gdy użytkownik wypełni formularz na stronie WWW i go prześle, przeglądarka wyśle żądanie POST. Ponieważ żądania POST zawierają informacje, które trzeba przesłać na serwer, przeglądarka wysyła je w *ciele żądania*, za nagłówkami POST.

W rozdziale 8. dowiesz się, dlaczego użycie metody POST zamiast GET jest takie ważne podczas wysyłania danych na serwer. Witryny WWW, które błędnie wykorzystują żądania GET do wykonywania zadań innych niż pobieranie zasobów, są narażone na ataki typu cross-site request forgery.

Podczas tworzenia witryny WWW możesz się też zetknąć z żądaniami PUT, PATCH i DELETE. Służą one odpowiednio do przesyłania, edytowania i usuwania zasobów na serwerze i zwykle są wywoływane przez kod JavaScriptu strony WWW. W tabeli 2.1 zebrano kilka innych wartych poznania metod.

Gdy serwer WWW otrzyma żądanie HTTP, wysyła odpowiedź agentowi użytkownika. Sprawdźmy, jaką strukturę mają odpowiedzi.

Tabela 2.1. Mniej znane metody HTTP

Metoda HTTP	Funkcja i implementacja
HEAD	Żądanie HEAD pobiera te same informacje co żądanie GET, ale informuje serwer, aby zwrócił odpowiedź bez ciała (innymi słowy: bez przydatnej części). Jeśli zaimplementujesz na serwerze WWW metodę GET, serwer automatycznie będzie odpowiadał także na żądania HEAD.
CONNECT	CONNECT inicjuje komunikację dwukierunkową. Użyj go w kodzie klienckim protokołu HTTP, aby umożliwić łączenie się poprzez proxy.
OPTIONS	Za pomocą żądania OPTIONS agent użytkownika może uzyskać informacje o innych metodach obsługiwanych przez zasób. Serwer WWW zwykle odpowiada na żądania OPTIONS, podając metody, które zaimplementowałeś.
TRACE	Odpowiedź na żądanie TRACE zawiera dokładną kopię oryginalnego żądania HTTP, dzięki czemu klient może sprawdzić, jakie zmiany (o ile takowe wystąpią) zostały wprowadzone przez serwery pośredniczące. Wydaje się, że jest to pożyteczny mechanizm, ale zwykle zaleca się wyłączenie żądań TRACE na serwerze WWW, ponieważ mogą stanowić lukę w zabezpieczeniach. Za ich pomocą można np. wstrzyknąć złośliwy kod JavaScriptu na stronę, aby uzyskać dostęp do danych z obiektów cookie, które zostały celowo zablokowane przed dostępem z poziomu kodu JavaScriptu.

Odpowiedzi HTTP

Odpowiedzi HTTP odsyłane przez serwer WWW zaczynają się od opisu protokołu, trzyznakowego *kodu stanu* i zwykle *opisu stanu*, informującego, czy żądanie można spełnić. Odpowiedź zawiera także nagłówki z metadanymi, które informują przeglądarkę, jak traktować treść odpowiedzi. Większość odpowiedzi zawiera również ciało, w którym znajduje się żądany zasób. Listing 2.2 przedstawia zawartość prostej odpowiedzi HTTP.

Listing 2.2. Odpowiedź HTTP zwrócona przez stronę example.com, najmniej ciekawą stronę internetową na świecie

```
HTTP/1.1❶ 200❷ OK❸
❹ Content-Encoding: gzip
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Content-Length: 606

❺ <!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1" />
  ❻ <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI",
        ↪"Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

❼ <body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

Odpowiedź zaczyna się od opisu protokołu ❶, kodu stanu ❷ oraz opisu stanu ❸. Kody stanu w formacie *2xx* oznaczają, że żądanie zostało zinterpretowane, zaakceptowane i została wysłana odpowiedź. Kody w formacie *3xx* przekierowują klienta na inny adres URL. Kody w postaci *4xx* oznaczają błąd klienta i są zwracane, gdy przeglądarka wygeneruje niepoprawne żądanie. (Z tej grupy błędów najczęściej występuje błąd HTTP 404 Not Found). Kody w formacie *5xx* oznaczają błąd serwera i występują, gdy żądanie jest poprawne, ale serwer nie może spełnić żądania.

Następnie znajdują się nagłówki HTTP ❹. Prawie wszystkie odpowiedzi HTTP zawierają nagłówek Content-Type, który informuje o typie zwracanych danych. Odpowiedzi na żądania GET zwykle zawierają również nagłówek Cache-Control informujący, że klient powinien zapisać lokalnie bardzo duże zasoby (np. obrazy).

Jeśli serwer może zwrócić odpowiedź HTTP, jej ciało zawiera zasób, który klient próbuje pobrać — zwykle jest to tekst w formacie *HyperText Markup Language* (*HTML*) ❺, opisujący strukturę żądanej strony WWW. W tym przypadku odpowiedź zawiera informacje o stylach ❻, a także treść strony ❼. Ciała odpowiedzi mogą też zawierać kod JavaScriptu, style CSS (Cascading Style Sheets) definiujące wygląd dokumentu HTML-a lub dane binarne.

Połączenia stanowe

Serwery WWW zwykle obsługują wiele agentów użytkowników jednocześnie, ale protokół HTTP nie zawiera żadnych wytycznych pozwalających ustalić, które żądanie pochodzi od określonego agenta. Na wczesnym etapie rozwoju internetu nie stanowiło to problemu, ponieważ strony WWW służyły przeważnie tylko do odczytu. Jednak nowoczesne serwisy WWW zwykle umożliwiają użytkownikom logowanie się i śledzą ich aktywność podczas wizyt i interakcji z różnymi stronami. Aby to umożliwić, komunikacja HTTP musi uwzględniać stan. Połączenie lub komunikacja między klientem a serwerem jest *stanowa* wtedy, gdy klient i serwer przeprowadzają procedurę uzgodnienia, tzw. handshake, i kontynuują wysyłanie pakietów w obydwu kierunkach, aż jedna ze stron zdecyduje się zakończyć połączenie.

Aby serwer WWW mógł śledzić, któremu użytkownikowi odsyła odpowiedź na określone żądanie, a tym samym prowadzić stanową komunikację HTTP, musi zaimplementować mechanizm śledzenia agentów użytkowników w miarę otrzymywania kolejnych żądań. Cała konwersacja między określonym agentem użytkownika a serwerem WWW jest określana mianem *sesji HTTP*. Najpopularniejszy sposób śledzenia stanu sesji na serwerze polega na odesłaniu nagłówka Set-Cookie w początkowej odpowiedzi HTTP. Dzięki temu agent użytkownika odbierający odpowiedź wie, że musi zapisać obiekt *cookie*, czyli mały fragment danych tekstowych przypisanych do określonej domeny WWW. Podczas wysyłania następnych żądań HTTP do tego samego serwera WWW agent użytkownika umieści te same dane w nagłówku Cookie. Jeśli mechanizm ten jest poprawnie zaimplementowany, obiekt cookie przesyłany podczas konwersacji w unikalny sposób identyfikuje agenta użytkownika, tworząc tym samym sesję HTTP.

Informacje o sesji zawarte w obiektach cookie są łakomym kąskiem dla hakerów. Jeśli atakujący ukradnie cookie innego użytkownika, może się pod niego podszywać w serwisie WWW. Podobnie jeśli z powodzeniem przekona witrynę do zaakceptowania sfałszowanego obiektu cookie, może się podszyć pod dowolnego użytkownika. W rozdziale 10. omawiam różne metody kradzieży i fałszowania obiektów cookie.

Szyfrowanie

Gdy wynaleziono sieć WWW, żądania i odpowiedzi HTTP były wysyłane zwykłym tekstem, co oznaczało, że mógł je odczytać każdy, komu udało się przechwycić pakiety danych; tego typu mechanizm nosi nazwę *ataku man-in-the-middle*. Ponieważ we współczesnej sieci WWW popularna jest wymiana prywatnych wiadomości oraz dokonywanie transakcji online, serwery WWW i przeglądarki chronią użytkowników przed tego typu atakami, stosując *szyfrowanie*. Szyfrując dane podczas transferu, ukrywają zawartość wiadomości przed niepożądanym wzrokiem.

Aby zabezpieczyć komunikację, serwery WWW i przeglądarki wysyłają żądania i odpowiedzi, korzystając z mechanizmu *Transport Layer Security* (*TLS*), czyli metody szyfrowania zapewniającej prywatność i integralność danych. TLS gwarantuje, że pakietów przechwyconych przez niepożądane podmioty nie będzie można rozszyfrować bez znajomości odpowiednich kluczy. Gwarantuje też wykrycie każdej próby manipulowania zawartością pakietów, co zapewnia integralność danych.

Komunikację HTTP odbywającą się za pośrednictwem TLS określa się mianem *HTTP Secure* (*HTTPS*). HTTPS wymaga od klienta i serwera wykonania procedury uzgodnienia, tzw. *TLS handshake*, podczas której obydwie strony ustalają metodę szyfrowania (szyfr) oraz wymieniają klucze szyfrujące. Po zakończeniu tej procedury wszystkie następne komunikaty (zarówno żądania, jak i odpowiedzi) będą nieczytelne dla innych użytkowników sieci.

Szyfrowanie jest złożonym zagadnieniem, jednak kluczowym dla zapewnienia bezpieczeństwa witryn internetowych. W rozdziale 13. opisuję, jak włączyć szyfrowanie dla witryny.

Podsumowanie

W tym rozdziale poznałeś kulisy działania internetu. TCP umożliwia niezawodną komunikację między komputerami połączonymi z internetem posiadającymi adresy IP. Domain Name System jest źródłem czytelnych dla ludzi aliasów adresów IP. Protokół HTTP opiera się na protokole TCP i umożliwia wysyłanie żądań HTTP przez agentów użytkowników (np. przeglądarki internetowe) na serwery WWW, które następnie odsyłają odpowiedzi HTTP. Każde żądanie jest przesyłane na określony adres URL. Poznałeś też różne typy metod HTTP. Serwery WWW zwracają kody stanu i odsyłają obiekty cookie, aby zainicjalizować połączenia stanowe. Aby zabezpieczyć komunikację między agentem użytkownika a serwerem WWW, można skorzystać z szyfrowania (za pośrednictwem protokołu HTTPS).

W następnym rozdziale dowiesz się, co się dzieje, gdy przeglądarka WWW otrzymuje odpowiedź HTTP. Poznasz tajniki renderowania strony WWW oraz sprawdzisz, jak działania użytkownika mogą wygenerować większą liczbę żądań HTTP.


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

NIE JEST ZA PÓŹNO. CHYBA ŻE JUŻ JEST...

Niemal każdego miesiąca słyszymy o spektakularnych atakach hakerskich. Konsekwencje? Straty finansowe, poważny uszczerbek na wizerunku, a nawet zagrożenie bezpieczeństwa publicznego. Wielokierunkowa i chaotyczna ewolucja technologii internetowych, łatwy dostęp do kodów źródłowych i aktywna społeczność zmotywowanych hakerów sprawiają, że uzyskanie wysokiego standardu bezpieczeństwa aplikacji internetowej wydaje się niemożliwe do osiągnięcia. Skoro ofiarami przestępców padają wielkie korporacje i instytucje rządowe, to jakie szanse w tym wyścigu zbrojeń ma zwykły programista?

To książka przeznaczona dla programistów o różnym stopniu zaawansowania. Gruntownie wyjaśnia charakter wszystkich istotnych zagrożeń i przedstawia zasady zapewniania bezpieczeństwa aplikacji internetowych. Opisuje także przykłady rzeczywistych ataków i mechanizmy wykorzystania luk w zabezpieczeniach. Zaprezentowane treści zostały wzbogacone dokładnie wyjaśnionym kodem, pokazano tu również, jak należy naprawiać opisane luki. Nawet jeśli jesteś wyjadaczem w dziedzinie kodowania, prędko się zorientujesz, czego jeszcze nie wiesz, i dzięki lekturze uzupełnisz wiedzę, by sprawnie wdrożyć najlepsze praktyki bezpieczeństwa.

Co ważne, autor nie ogranicza się do jednego języka programowania — uwzględnia zalecenia dotyczące bezpieczeństwa we wszystkich najważniejszych językach.

Ta książka pomoże Ci:

- zapobiegać wstrzykiwaniu kodu SQL, szkodliwego JavaScriptu i atakom typu *cross-site*
- chronić konta użytkowników przed kradzieżą haseł i sesji lub eskalacją uprawnień
- zaimplementować szyfrowanie i usunąć luki ze starszego kodu
- zapobiegać ujawnianiu luk w zabezpieczeniach
- chronić się przed zaawansowanymi atakami typu *malvertising* i *denial-of-service*

Malcolm McDonald przez dwadzieścia lat pisał kod dla firm finansowych i start-upów. Prowadził też cenione szkolenia na temat luk w zabezpieczeniach. Jest twórcą witryny *hacksplaining.com*, w której publikuje materiały szkoleniowe dotyczące bezpieczeństwa w tworzeniu aplikacji internetowych. Mieszka w Oakland w stanie Kalifornia z żoną i kotem.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

ISBN 978-83-283-7803-2
9 788328 378032

Cena: 59,00 zł
no starch press