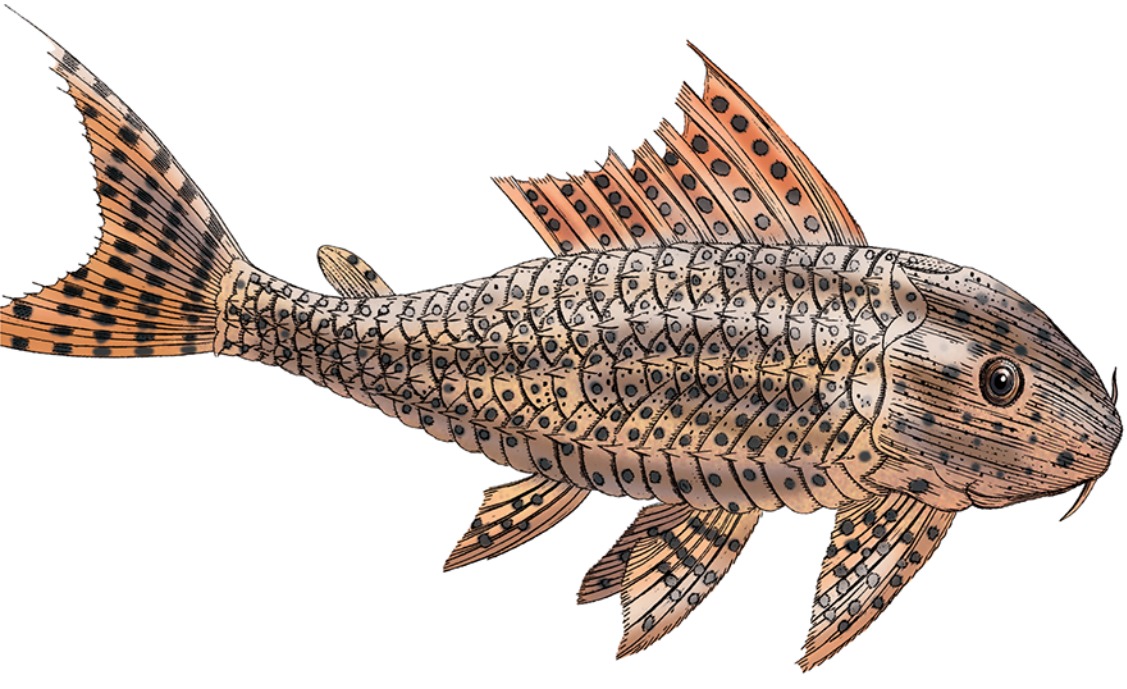
O'REILLY®

Kontenery
Bezpieczne wdrożenia
Podstawowe koncepcje i technologie



Helion

Liz Rice

Tytuł oryginału: Container Security: Fundamental Technology Concepts that Protect Containerized Applications

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-7228-3

Helion SA
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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/konten.zip

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

Printed in Poland.

Spis treści

ROZDZIAŁ 1.

Zagrożenia związane z kontenerami

W ostatnich latach można zauważyć zwiększone zainteresowanie kontenerami. Wprawdzie koncepcja kontenera istniała już wiele lat przed pojawieniem się Dockera w 2013 roku, ale większość obserwatorów zgodzi się z tym, że to łatwe w użyciu narzędzia powłoki zaoferowane przez Dockera przyczyniły się do popularyzacji kontenerów w społeczności programistów.

Kontener oferuje wiele zalet wymienionych w oryginalnym sloganie Dockera, pozwala na „jednokrotne utworzenie rozwiązania, które następnie może być uruchamiane gdziekolwiek". Jest to możliwe za sprawą połączenia aplikacji z jej wszystkimi zależnościami oraz odizolowania aplikacji od urządzenia, w którym jest uruchamiana. Aplikacja w kontenerze ma wszystko to, co jest potrzebne do działania. Bardzo łatwo przygotować ją w postaci kontenera, który następnie w ten sam sposób będzie działał w zarówno Twoim laptopie, jak i serwerze umieszczonym w centrum danych.

Efektem domina wynikającym z tej izolacji jest możliwość jednoczesnego uruchamiania wielu różnych kontenerów, które nie będą sobie wzajemnie przeszkadzały. Przed powstaniem kontenerów bardzo łatwo można było doprowadzić do tzw. piekła zależności, z którym mieliśmy do czynienia, gdy dwie aplikacje wymagały odmiennych wersji tych samych pakietów. Najłatwiejszym rozwiązaniem tego problemu było uruchamianie aplikacji w oddzielnych komputerach. Kiedy korzystamy z kontenerów, zależności są odizolowane od siebie, co z kolei znacznie ułatwia uruchamianie wielu aplikacji w tym samym serwerze. Użytkownicy bardzo szybko przekonali się o możliwości wykorzystania zalet kontenerów do uruchamiania wielu aplikacji w tym samym komputerze gospodarza (niezależnie od tego, czy to jest maszyna wirtualna, czy fizyczny serwer) bez konieczności przejmowania się zależnościami.

Następnym logicznym krokiem było rozpowszechnienie skonteneryzowanych aplikacji w klastrze serwerów. Orkiestratory, takie jak Kubernetes, automatyzują ten proces, uwalniając tym samym programistów od konieczności ręcznej instalacji aplikacji w określonym komputerze. Wystarczy wskazać kontenery, które mają być uruchomione, a orkiestrator odszuka odpowiednie położenie dla każdego z nich.

Z perspektywy zapewnienia bezpieczeństwa w środowisku aplikacji działających w kontenerach wiele aspektów nie różni się niczym od używanych w tradycyjnych wdrożeniach. Na świecie istnieje wiele osób atakujących, które chcą ukraść dane, zmodyfikować sposób działania systemu bądź też wykorzystać zasoby urządzeń użytkowników do wydobywania kryptowalut. Te niebezpieczeństwa nie przestaną zagrażać aplikacjom przeniesionym do kontenerów. Jednak kontenery zmieniają sposób uruchamiania aplikacji, co z kolei oznacza, że stają się narażone na kolejne rodzaje niebezpieczeństw.

Ryzyko, zagrożenie i środki ostrożności

Ryzyko to potencjalny problem i powodowane przez niego efekty, gdy dany problem wystąpi.

Zagrożenie to ścieżka prowadząca do danego problemu.

Środki ostrożności to przeciwdziałanie zagrożeniu — coś, co można zrobić, aby uniknąć zagrożenia lub przynajmniej zmniejszyć jego negatywne skutki.

Przykładowo istnieje ryzyko, że ktoś ukradnie z Twojego domu kluczyki do samochodu i tym samym odjedzie Twoim pojazdem w siną dal. W takim przypadku zagrożenie to różne sposoby kradzieży tych kluczyków: wybicie szyby i zabranie kluczyków, zapukanie do drzwi i zagadanie Cię, gdy w tym samym czasie wspólnik dostaje się do domu i szybko kradnie kluczyki itd. Środki ostrożności dla wymienionych zagrożeń to unikanie pozostawiania kluczyków do samochodu w widocznym miejscu.

Ryzyko zmienia się znacznie w zależności od organizacji. W banku przechowującym pieniądze klientów największym ryzykiem jest kradzież tych pieniędzy. Z kolei firma prowadząca sprzedaż internetową jest narażona na ryzyko przeprowadzania fałszywych transakcji. Natomiast osoba prowadząca witrynę internetową zawierającą blog może obawiać się, że ktoś włamie się do witryny, podszyje pod autora i będzie publikował nieodpowiednie treści, np. komentarze. Ponieważ dotyczące prywatności regulacje prawne są różne w poszczególnych krajach, ryzyko wycieku prywatnych danych użytkowników zależy od położenia geograficznego. W wielu krajach ryzykiem jest „jedynie" utrata reputacji, natomiast w Unii Europejskiej obowiązuje dyrektywa dotycząca ochrony danych osobowych (RODO), zgodnie z którą firma może zostać ukarana grzywną wynoszącą do 4% całkowitych przychodów firmy (*https://www.csoonline.com/article/3518370/the-biggest-ico-fines-for-data-protection-and-gdpr-breaches.html*).

Z powodu znacznych różnic w ryzyku względna waga poszczególnych zagrożeń również się zmienia, podobnie jak i dostępne środki ostrożności. Framework zarządzania ryzykiem to proces pozwalający na potraktowanie ryzyka w sposób systematyczny, oszacowanie możliwych zagrożeń, ułożenie ich według wagi oraz zdefiniowanie środków ostrożności.

Modelowanie ryzyka to proces identyfikacji potencjalnych zagrożeń dla systemu. Dzięki systematycznemu sprawdzaniu komponentów systemu i możliwych trybów ataku model zagrożenia może pomóc w ustaleniu miejsc, w których system jest najbardziej narażony na atak.

Nie istnieje jeden uniwersalny model zagrożenia, ponieważ zależy ono od ryzyka, konkretnego środowiska, organizacji oraz uruchomionych w nim aplikacji. Mimo to, istnieje możliwość przygotowania listy potencjalnych zagrożeń, które często występują w większości wdrożeń kontenerów, o ile nie we wszystkich.

Model zagrożenia kontenera

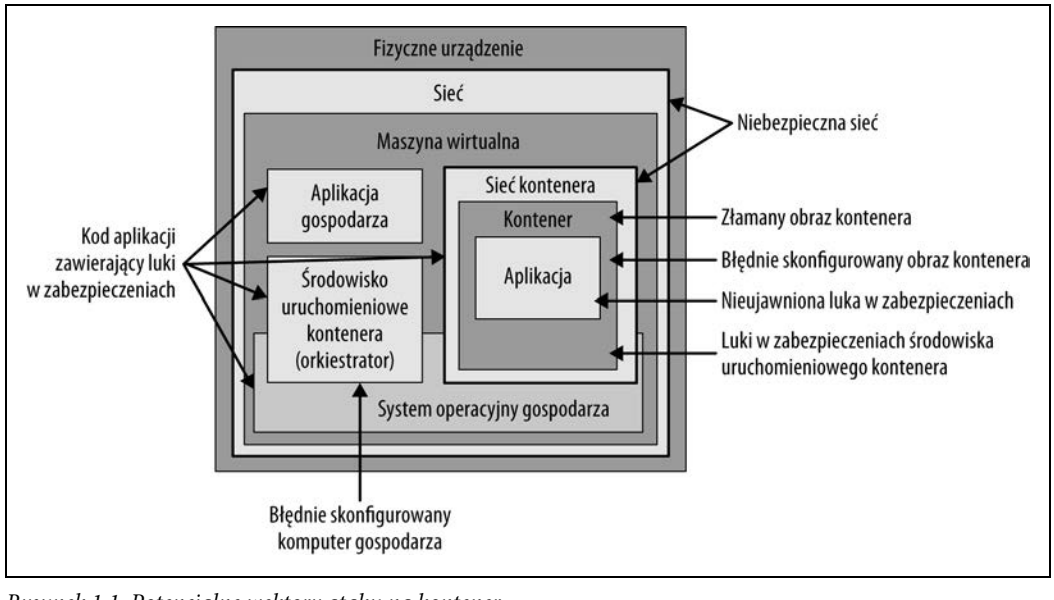
Jednym ze sposobów analizy modelu zagrożenia jest rozważenie związanych z tym działań. Możemy do nich zaliczyć tutaj wymienione:

- *osoba atakująca, która z zewnątrz* próbuje uzyskać dostęp do wdrożenia;
- *osoba atakująca, która z wewnątrz* zapewnia sobie dostęp do pewnych elementów wdrożenia;
- *złośliwie działająca osoba znajdująca się wewnątrz organizacji*, np. programista lub administrator, mająca uprawnienia pozwalające jej na uzyskanie dostępu do wdrożenia;
- *nieuważnie działająca osoba znajdująca się wewnątrz organizacji* może przypadkowo spowodować problemy;
- *proces aplikacji*, który nie jest przeznaczony do złamania systemu, może w sposób programowy mieć dostęp do systemu.

Każdy z użytkowników ma określony zestaw uprawnień, o czym trzeba pamiętać.

- Jaki poziom dostępu zapewniają uprawnienia? Czy przykładowo dana osoba ma dostęp do kont użytkowników w komputerze gospodarza, w którym będzie przeprowadzone wdrożenie aplikacji?
- Jakie uprawnienia w systemie ma dana osoba? W przypadku Kubernetes może odwoływać się do opartych na roli ustawień dla poszczególnych użytkowników, a także dla użytkowników anonimowych.
- Jak szeroki dostęp do sieci ma dana osoba? Które przykładowo elementy systemu znajdują się w ramach wirtualnej chmury prywatnej (ang. *virtual private cloud*, VPC)?

Istnieje wiele potencjalnych sposobów przeprowadzania ataku na skonteneryzowane wdrożenie. Jednym z rozwiązań pozwalających na mapowanie wspomnianych ataków jest przeanalizowanie potencjalnych wektorów ataku na poszczególnych etapach cyklu życiowego kontenera, co pokazałam na rysunku 1.1.



Rysunek 1.1. Potencjalne wektory ataku na kontener

Kod aplikacji zawierający luki w zabezpieczeniach

Cykl życiowy rozpoczyna się od kodu aplikacji tworzonego przez programistę. Ten kod i jego zależności zewnętrzne mogą zawierać błędy określane mianem luk w zabezpieczeniach. Opublikowane zostały informacje o tysiącach znalezionych luk w zabezpieczeniach, które osoba atakująca może wykorzystać, jeśli dana luka istnieje w aplikacji. Najlepszym sposobem uniknięcia uruchomienia kontenera z doskonale znanymi lukami w zabezpieczeniach jest skanowanie obrazów, czego przykład przedstawię w rozdziale 7. Nie jest to jednorazowa operacja, ponieważ nowe luki w zabezpieczeniach są nieustannie odkrywane w istniejącym kodzie. Proces skanowania musi również sprawdzić, czy w kontenerze zostały wykorzystane nieaktualne pakiety, które wymagają aktualizacji ze względu na istnienie ich nowszych wersji zawierających poprawki bezpieczeństwa. Część skanerów pozwala również na wykrycie oprogramowania typu malware znajdującego się w obrazie kontenera.

Błędnie skonfigurowany obraz kontenera

Kod po utworzeniu zostaje umieszczony w obrazie kontenera. Gdy konfigurujesz sposób przygotowania kontenera, masz wiele okazji do popełnienia błędów, które następnie mogą być wykorzystane do przeprowadzenia skutecznego ataku na działający kontener. Przykładem może być uruchomienie kontenera z uprawnieniami użytkownika `root`, co oznacza przydzielenie kontenerowi znacznie większych uprawnień, niż faktycznie są wymagane. Więcej informacji na temat znajdziesz w rozdziale 6.

Atak na komputer, w którym jest przygotowywany obraz

Jeżeli osoba atakująca będzie mogła zmodyfikować sposób utworzenia obrazu kontenera lub wpłynąć na niego, wówczas zyska możliwość wstawienia kodu o złośliwym działaniu, który następnie trafi do środowiska produkcyjnego i tam zostanie uruchomiony. Ponadto znalezienie punktu zaczepienia w środowisku tworzenia obrazu kontenera będzie pierwszym krokiem pozwalającym na udane włamanie do środowiska produkcyjnego. Ten temat również będzie poruszony w rozdziale 6.

Atak na łańcuch dostawców

Obraz kontenera po utworzeniu zostaje umieszczony w rejestrze, z którego jest następnie pobierany (ang. *pull*) tam, gdzie ma być uruchomiony. Czy można mieć pewność, że pobrany obraz jest dokładnie tym samym obrazem, który wcześniej został umieszczony w rejestrze? Czy ten obraz na pewno nie został zmodyfikowany? Jeżeli osoba atakująca może zastąpić obraz innym lub zmodyfikować go między etapami utworzenia obrazu i jego wdrożenia, wówczas będzie mogła w Twoim środowisku wdrożenia uruchomić dowolny kod. To kolejny temat omówiony w rozdziale 6.

Błędnie skonfigurowany kontener

Jak się dowiesz w rozdziale 9., istnieje niebezpieczeństwo uruchomienia kontenera wraz z ustawieniami dającymi mu niepotrzebne i prawdopodobnie nieplanowane uprawnienia. Jeżeli pobrałeś z internetu plik konfiguracyjny YAML, nie uruchamiaj go bez wcześniejszego dokładnego przeanalizowania znajdujących się w nim ustawień.

Komputer gospodarza zawierający luki w zabezpieczeniach

Kontenery mogą być uruchamiane w maszynach wirtualnych i trzeba zagwarantować, że w komputerze gospodarza nie działa kod zawierający luki w zabezpieczeniach (np. starsza wersja orkiestratora wraz z doskonale znanymi lukami w zabezpieczeniach). Dobrym rozwiązaniem jest ograniczenie do absolutnego minimum oprogramowania zainstalowanego w komputerze gospodarza, aby w ten sposób zmniejszyć płaszczyznę ataku. Ponadto komputer gospodarza musi być skonfigurowany zgodnie z najlepszymi praktykami w zakresie zapewnienia bezpieczeństwa. Kwestie te zostaną dokładniej omówione w rozdziale 4.

Ujawnione dane poufne

Aby kod aplikacji mógł prowadzić komunikację z komponentami systemu, często musi mieć dostęp do danych uwierzytelniających, tokenów lub haseł. We wdrożeniach opartych na kontenerach wymienione dane poufne muszą być przekazane do kodu umieszczonego w kontenerze. Jak się dowiesz w rozdziale 12., można to zrobić na wiele różnych sposobów, które różnią się poziomem zapewnienia bezpieczeństwa.

Niebezpieczna sieć

Ogólnie rzecz biorąc, kontenery muszą mieć możliwość komunikacji między sobą lub ze światem zewnętrznym. W rozdziale 10. pokażę sposób działania sieci w kontenerach, natomiast w rozdziale 11. wyjaśnię, jak zdefiniować bezpieczne połączenia między kontenerami.

Luki w zabezpieczeniach środowiska uruchomieniowego kontenera

Do często używanych środowisk uruchomieniowych kontenerów zaliczamy `containerd` i `CRI-O`, które są dość dobrze zabezpieczone. Jednak — mimo to — istnieje niebezpieczeństwo, że zawierają jeszcze nieodkryte błędy pozwalające, aby uruchomiony w kontenerze kod o złośliwym działaniu mógł dostać się do systemu operacyjnego gospodarza. Jedna z takich sytuacji, określanych mianem *runcescape* (*https://thenewstack.io/what-you-need-to-know-about-the-runc-container-escape-vulnerability/*), ujrzała światło dzienne w 2019 roku. W rozdziale 4. znajdziesz informacje dotyczące izolacji, dzięki której kod aplikacji powinien działać jedynie w kontenerze. W przypadku części aplikacji skutki uruchomienia w systemie komputera gospodarza kodu o złośliwym działaniu mogą być na tyle poważne, że warto rozważyć zastosowanie silniejszych mechanizmów izolacji, które przedstawię w rozdziale 8.

Mamy jeszcze inne płaszczyzny ataków, które nie zostały omówione w książce.

- Kod źródłowy jest przechowywany w repozytoriach, co — ogólnie rzecz biorąc — jest wygodnym rozwiązaniem, choć jednocześnie tworzy kolejną płaszczyznę ataku na aplikację. Trzeba więc zagwarantować odpowiednią kontrolę w zakresie dostępu, jaki użytkownicy mają do repozytoriów kodu źródłowego.
- Komputery gospodarzy się połączone. W celu zapewnienia bezpieczeństwa często wykorzystują VPC i zwykle mają połączenie z internetem. Podobnie jak w tradycyjnych wdrożeniach, także podczas pracy z kontenerami konieczna jest ochrona komputerów gospodarzy (lub maszyn wirtualnych) przed zagrożeniami związanymi z dostępem do tych urządzeń. Bezpieczna konfiguracja sieci, stosowanie zapór sieciowych oraz identyfikacja dostępu do zasobów i zarządzanie nimi wciąż mają zastosowanie we wdrożeniach do natywnych chmur, podobnie jak ma to miejsce w tradycyjnych wdrożeniach.

- Działanie kontenerów jest zwykle nadzorowane przez orkiestrator — w obecnie stosowanych wdrożeniach to najczęściej Kubernetes, choć dostępne są także inne orkiestratory, np. Docker Swarm i Hashicorp Nomad. Jeżeli orkiestrator został skonfigurowany w sposób niezapewniający bezpieczeństwa lub brakuje efektywnej kontroli nad udzielaniem uprawnień administratora, wówczas osoba atakująca zyskuje kolejną płaszczyznę ataku na wdrożoną aplikację.

Więcej informacji na temat modeli zagrożenia we wdrożeniach Kubernetes znajdziesz w raporcie *Kubernetes Threat Model* (*https://github.com/kubernetes/community/blob/master/wg-security-audit/findings/Kubernetes Threat Model.pdf*) opracowanym przez fundację CNCF.

Grupa CNCF Financial User Group opublikowała artykuł zatytułowany *Kubernetes Attack Tree* (*https://github.com/cncf/financial-user-group/tree/master/projects/k8s-threat-model*) przygotowany na podstawie metodologii STRIDE (*https://en.wikipedia.org/wiki/STRIDE_(security)*).

Granice bezpieczeństwa

Granice bezpieczeństwa (czasami nazywane również granicami zaufania) występują między elementami systemu, np. gdy przejście między tymi elementami wymaga odmiennych zestawów uprawnień. Zdarza się, że te granice są definiowane administracyjnie. Przykładowo w systemach z rodziny Linux administrator systemu może modyfikować granice bezpieczeństwa przez zdefiniowanie, do których plików użytkownik ma dostęp. (Odbywa się to za pomocą zmiany grupy, do której należy dany użytkownik). Jeżeli już zapomniałeś o uprawnieniach plików w systemie Linux, informacje zamieszczone w rozdziale 2. odświeżą Ci pamięć.

Kontener to rodzaj granicy bezpieczeństwa. Kod aplikacji ma działać w ramach kontenera i nie powinien mieć możliwości uzyskania dostępu do kodu lub danych na zewnątrz kontenera, o ile nie otrzyma takich uprawnień (np. dotyczących woluminu zamontowanego w kontenerze).

Im więcej granic bezpieczeństwa znajduje się między osobą atakującą i jej celem (takim jak np. dane Twoich klientów), tym trudniej będzie przeprowadzić skuteczny atak na dany cel.

Omówione wcześniej w tym rozdziale płaszczyzny ataku mogą być łączone, aby w ten sposób wykorzystać luki w zabezpieczeniach wielu różnych granic bezpieczeństwa. Przeanalizuj przedstawione tutaj przykłady.

- Z wykorzystaniem odkrytej luki w zabezpieczeniach zależności aplikacji osoba atakująca może zyskać możliwość zdalnego wykonania kodu w kontenerze.
- Przyjmujemy założenie, że skutecznie zaatakowany kontener nie ma bezpośredniego dostępu do jakichkolwiek wartościowych danych. W takim przypadku osoba atakująca musi znaleźć możliwość wydostania się z kontenera i przejścia do innego kontenera lub systemu komputera gospodarza. Luka w zabezpieczeniach środowiska uruchomieniowego kontenera okaże się jedną z takich możliwości, a niezapewniająca bezpieczeństwa konfiguracja to kolejna możliwość przedostania się z kontenera do gospodarza. Jeżeli osoba atakująca odkryje jedną z tych dróg, będzie mogła uzyskać dostęp do systemu operacyjnego gospodarza.

- Następnym krokiem jest szukanie sposobów na uzyskanie uprawnień użytkownika `root` w systemie komputera gospodarza. To może okazać się całkiem łatwym zadaniem, jeśli kod aplikacji został w kontenerze uruchomiony z uprawnieniami użytkownika `root`. Więcej informacji na temat przedstawię w rozdziale 4.
- Mając uprawnienia użytkownika `root` w komputerze gospodarza, osoba atakująca ma pełnię możliwości w tej maszynie, a tym samym dostęp do wszystkich uruchomionych w niej kontenerów.

Dodanie i wzmocnienie granic bezpieczeństwa w środowisku wdrożenia znacznie utrudni działania podejmowane przez agresora.

Ważnym aspektem modelu zagrożenia jest również uwzględnienie ewentualnego ataku w środowisku, w którym została uruchomiona aplikacja. We wdrożeniach w chmurze pewne zasoby mogą być współdzielone z innymi użytkownikami i ich aplikacjami. Współdzielenie zasobów maszyny jest określane mianem *wielodostępności* (ang. *multitenancy*); jest to czynnik o ważnym znaczeniu w modelu zagrożenia.

Wielodostępność

W środowisku wielodostępnym poszczególni użytkownicy, w języku angielskim czasem określani terminem *tenant*, wykonują swoje zadania we współdzielonej maszynie. (Z pojęciem wielodostępności możesz się spotkać także w kontekście aplikacji, w takim przypadku odnosi się do wielu użytkowników korzystających z tego samego egzemplarza oprogramowania. Na potrzeby materiału przedstawionego w książce przyjęłam założenie, że współdzielona jest tylko fizyczna maszyna). W zależności od tego, kto wykonuje zadania we współdzielonej maszynie, a także od liczby poszczególnych użytkowników i poziomu zaufania między nimi, może okazać się konieczne zdefiniowanie silniejszych granic, które uniemożliwią użytkownikom przeszkadzanie sobie nawzajem.

Wielodostępność to koncepcja istniejąca od czasów komputerów typu mainframe w latach 60. ubiegłego wieku, kiedy to klienci płacili za czas pracy procesora, pamięć operacyjną i pamięć masową we współdzielonych maszynach. Podobne rozwiązanie jest stosowane w obecnych chmurach publicznych — takich jak Amazon AWS, Microsoft Azure i Google Cloud Platform — w których klient płaci za czas pracy procesora, pamięć operacyjną, pamięć masową oraz inną funkcjonalność i zarządzane usługi. Od chwili pojawienia się chmury Amazon AWS w 2006 roku istnieje możliwość wypożyczania egzemplarzy maszyn wirtualnych działających w serwerach centrów danych umieszczonych na całym świecie. W fizycznym komputerze może być uruchomionych wiele maszyn wirtualnych (ang. *virtual machines*), a klient korzystający z chmury udostępniającej te maszyny wirtualne nawet nie będzie wiedział, kto używa innych maszyn wirtualnych uruchomionych w tym samym fizycznym serwerze.

Maszyny współdzielone

Zdarzają się sytuacje, w których pojedynczy komputer (lub maszyna wirtualna) działający pod kontrolą systemu operacyjnego Linux będzie współdzielony przez wielu użytkowników. Takie rozwiązanie jest często spotykane, np. w instytucjach edukacyjnych, i stanowi dobry przykład prawdziwej

wielodostępności w sytuacjach, gdy użytkownicy nie ufają sobie nawzajem i — szczerze mówiąc — administratorzy nie ufają użytkownikom. W takim środowisku mechanizmy kontroli dostępu w systemie Linux są stosowane do nakładania ścisłych ograniczeń na możliwości oferowane użytkownikom. Każdy użytkownik otrzymuje własny identyfikator, a wspomniane mechanizmy gwarantują, że będzie mógł modyfikować jedynie pliki znajdujące się w jego katalogu domowym. Czy możesz wyobrazić sobie chaos, który powstałby, gdyby uczniowie w szkole mogli odczytywać lub — co gorsza — modyfikować pliki należące do innych uczniów?

Jak się dowiesz w rozdziale 4., wszystkie kontenery uruchomione w tym samym współdzielonym komputerze używają dokładnie tego samego jądra systemu gospodarza. Jeżeli w komputerze został uruchomiony demon Dockera, każdy użytkownik mający możliwość wydania polecenia `docker` praktycznie otrzymuje uprawnienia użytkownika `root`. Dlatego też administrator nie zezwoli na wydawanie wymienionego polecenia tym użytkownikom, którym nie ufa.

W środowiskach przedsiębiorstw, a dokładnie w natywnych środowiskach chmury, istnieje znacznie mniejsze prawdopodobieństwo spotkania takiego rodzaju maszyny współdzielonej. Zamiast tego użytkownicy (lub zespoły ufających sobie użytkowników) zwykle będą korzystali z własnych zasobów, zaalokowanych w postaci maszyn wirtualnych.

Wirtualizacja

Ogólnie rzecz biorąc, maszyny wirtualne są uznawane za dość silnie odizolowane od siebie, co wcale nie oznacza, że nie mogą sobie wzajemnie przeszkadzać. Więcej informacji na temat izolacji stosowanej w maszynach wirtualnych znajdziesz w rozdziale 5. Zgodnie z ogólnie przyjętą definicją, *https://en.wikipedia.org/wiki/Multitenancy*, wirtualizacja w ogóle nie jest uznawana za wielodostępność. Z wielodostępnością mamy do czynienia, gdy różne grupy osób współdzielą pojedynczy egzemplarz tego samego oprogramowania. W przypadku wirtualizacji użytkownicy nie mają dostępu do hipernadzorcy (ang. *hypervisor*) zarządzającego maszynami wirtualnymi, więc nie ma tutaj mowy o współdzieleniu jakiegokolwiek oprogramowania.

To oczywiście nie oznacza, że izolacja zachodząca między maszynami wirtualnymi jest doskonała. Wielokrotnie można było się spotkać z narzekaniami użytkowników na „hałaśliwych sąsiadów", gdy na skutek współdzielenia fizycznego komputera z innymi użytkownikami dochodziło do nieoczekiwanych zmian w wydajności działania. Firma Netflix korzysta z chmury publicznej od samego początku jej istnienia. W sekcji *Co-tenancy is hard* posta bloga opublikowanego w 2010 roku (*https://netflixtechblog.com/5-lessons-weve-learned-using-aws-1f2a28588e4c*) przyznała się do tworzenia systemów, które celowo mogą porzucać podzadanie, jeśli wydajność ich działania okazuje się zbyt wolna. W ostatnim czasie można się spotkać z opiniami, że problem „hałaśliwych sąsiadów" praktycznie nie istnieje (*https://www.infoworld.com/article/3073503/debunking-the-clouds-noisy-neighbor-myth.html*).

Zdarzały się przypadki, gdy luki w zabezpieczeniach aplikacji doprowadzały do naruszenia granic między maszynami wirtualnymi.

Dla pewnych aplikacji i organizacji (zwłaszcza rządowych, finansowych i związanych z ochroną zdrowia) konsekwencje wynikające z udanego wykorzystania luki w zabezpieczeniach uzasadniają zastosowanie fizycznej separacji. Można skorzystać z chmury prywatnej; centrum danych jest

wówczas uruchomione lub zarządzane przez dostawcę usługi, a klient ma gwarancję całkowitej izolacji danych. Chmura prywatna czasami oferuje dodatkowe funkcje zabezpieczeń w postaci np. dokładniejszego sprawdzania personelu, który ma dostęp do centrum danych.

Wielu dostawców chmury oferuje rozwiązania oparte na maszynach wirtualnych i jednocześnie gwarantuje, że maszyny należące do danego klienta są uruchamiane w oddzielnym fizycznym komputerze. Istnieje również możliwość wypożyczenia fizycznego komputera zarządzanego przez dostawcę chmury. W takich sytuacjach problem „hałaśliwych sąsiadów" w ogóle nie istnieje, a korzyścią dodatkową jest znacznie silniejsza izolacja między fizycznymi komputerami.

Niezależnie od przyjętego rozwiązania — wypożyczenie komputera fizycznego lub maszyn wirtualnych, użycie chmury lub własnych serwerów — jeśli korzystasz z kontenerów, musisz uwzględnić granice bezpieczeństwa między grupami użytkowników.

Wielodostępność w przypadku kontenerów

Jak się przekonasz w rozdziale 4., izolacja między kontenerami nie jest tak silna, jak izolacja między maszynami wirtualnymi. Wprawdzie wiele zależy od profilu ryzyka, ale istnieje znikome prawdopodobieństwo, że będziesz chciał używać kontenerów w tej samej maszynie, w której działają kontenery podmiotu, któremu nie ufasz.

Jeśli nawet wszystkie kontenery działające w Twoich maszynach są uruchamiane przez Ciebie lub osoby, którym w pełni ufasz, nadal możesz chcieć ograniczyć do minimum niebezpieczeństwo błędu człowieka i zagwarantować, że kontenery nie będą sobie wzajemnie przeszkadzały.

Do podzielenia używanego klastra maszyn według poszczególnych osób, zespołów lub aplikacji, w Kubernetes można stosować tzw. *przestrzenie nazw*.

Wyrażenie *przestrzenie nazw* zostało użyte tutaj na wyrost. W Kubernetes przestrzeń nazw to wysokiego poziomu abstrakcja, dzięki której w poszczególnych zasobach klastra można stosować odmienne poziomy dostępu Kubernetes. Z kolei w systemie operacyjnym z rodziny Linux przestrzeń nazw to działający na niskim poziomie mechanizm izolacji zasobów komputera dostępnych dla danego procesu. Więcej informacji na temat takiej przestrzeni nazw znajdziesz w rozdziale 4.

Użycie kontroli dostępu na podstawie roli (ang. *role-based access control*, RBAC) pozwala na ograniczenie możliwości użytkowników i komponentów podczas uzyskiwania dostępu do poszczególnych przestrzeni nazw Kubernetes. Omówienie szczegółów takiego rozwiązania wykracza poza zakres tematyczny tej książki. Chciałabym jednak podkreślić, że stosowanie technik RBAC w Kubernetes ma wpływ jedynie na działania przeprowadzane za pomocą API Kubernetes. Kontenery aplikacji w tzw. *podach* Kubernetes działających w tym samym komputerze gospodarza są chronione i, zgodnie z informacjami zamieszczonymi w książce, odizolowane od siebie nawet wtedy, kiedy znajdują się w odmiennych przestrzeniach nazw. Jeżeli osobie atakującej uda się opuścić kontener i przeniknąć do systemu komputera gospodarza, granice przestrzeni nazw w Kubernetes nie powodują żadnej różnicy w możliwościach, jakie ten atakujący będzie miał względem pozostałych kontenerów.

Egzemplarze kontenera

Usługi chmury, takie jak Amazon AWS, Microsoft Azure i Google Cloud Platform, oferują wiele tzw. *usług zarządzanych*, za pomocą których użytkownik może wypożyczyć oprogramowanie, pamięć masową oraz inne komponenty bez konieczności instalowania ich i zarządzania nimi. Klasycznym przykładem jest tutaj usługa RDS (ang. *relational database service*) firmy Amazon. Przy jej użyciu można bardzo łatwo tworzyć bazy danych wykorzystujące doskonale znane oprogramowanie, np. PostgreSQL, i przez jedynie zaznaczenie pola wyboru definiować tworzenie kopii zapasowej (oczywiście za odpowiednią opłatą).

Usługi zarządzane stanowią rozszerzenie świata kontenerów. Azure Container Instances i AWS Fargate to przykłady usług pozwalających na uruchamianie kontenerów bez konieczności przejmowania się komputerem (lub maszyną wirtualną), w którym te kontenery działają.

W ten sposób można uniknąć znacznego obciążenia związanego z zarządzaniem usługami, a jednocześnie zyskuje się możliwość łatwego skalowania wdrożenia na żądanie. Jednak — przynajmniej teoretycznie — egzemplarze kontenerów będą znajdowały się w tej samej maszynie wirtualnej, w której działają kontenery innych klientów dostawcy usług. Jeżeli masz pod tym względem wątpliwości, najlepiej zapytaj dostawcę usług.

W tym momencie powinieneś mieć świadomość istnienia wielu potencjalnych zagrożeń dla Twoich aplikacji uruchamianych w kontenerach. Zanim przejdziemy do pozostałej części książki, chciałabym przedstawić krótkie wprowadzenie do zasad zapewnienia bezpieczeństwa, które powinieneś znać podczas analizy narzędzi bezpieczeństwa i procesów koniecznych do zastosowania we wdrożeniu.

Zasady dotyczące zapewnienia bezpieczeństwa

Istnieje kilka ogólnych zasad zapewnienia bezpieczeństwa, których stosowanie jest ogólnie uznawane za rozsądne, niezależnie od chronionych informacji i danych.

Najmniejsze możliwe uprawnienia

Zgodnie z zasadą najmniejszych uprawnień należy użytkownikowi lub komponentowi ograniczyć dostęp do absolutnego minimum niezbędnego do wykonania danego zadania. Jeśli przykładowo masz mikrousługę przeprowadzającą operację wyszukiwania produktu w aplikacji typu e-commerce, zasada najmniejszych uprawnień sugeruje, że ta mikrousługa powinna mieć tylko uprawnienia pobierania informacji z bazy danych produktów. Taka usługa nie wymaga dostępu do informacji np. o użytkowniku lub płatności, a także nie potrzebuje uprawnień zapisu informacji o produkcie.

Głęboka defensywa

Jak się przekonasz, czytając książkę, istnieje wiele różnych sposobów, na jakie można poprawić bezpieczeństwo wdrożenia i uruchomionych w nim aplikacji. Zasada głębokiej defensywy oznacza, że należy stosować wiele linii obrony. Jeżeli osobie atakującej uda się złamać jedną linię obrony, istnienie kolejnej linii obrony uniemożliwi jej kradzież danych lub dokonanie szkód we wdrożeniu.

Ograniczenie płaszczyzny ataku

Zgodnie z ogólną regułą, im bardziej skomplikowany system, tym większe niebezpieczeństwo, że będzie można go skutecznie zaatakować. Uproszczenie systemu może wzmocnić jego odporność na atak. Mam tutaj na myśli następujące kwestie:

- zmniejszenie punktów dostępu poprzez użycie małych i prostych interfejsów wszędzie tam, gdzie to tylko jest możliwe;
- ograniczenie liczby użytkowników i komponentów, które mają dostęp do danej usługi;
- zmniejszenie do minimum ilości tworzonego kodu.

Ograniczenie pola rażenia

Koncepcja segmentowania mechanizmów bezpieczeństwa na mniejsze podkomponenty (czasami można się spotkać również z określeniem ich mianem „komórek") oznacza, że w najgorszym przypadku, jakim jest skuteczny atak, zasięg będzie ograniczony. Kontenery doskonale dopasowują się do tej zasady, ponieważ dzięki podziałowi architektury na wiele egzemplarzy mikrousług kontener może działać w charakterze granicy bezpieczeństwa.

Podział zadań

Z zasadami najmniejszych uprawnień i ograniczenia pola rażenia jest powiązana zasada maksymalnego podziału zadań. Wtedy poszczególni użytkownicy i komponenty otrzymają uprawnienia do jedynie jak najmniejszego możliwego fragmentu całego systemu. Takie podejście ogranicza potencjalne szkody, jakie mogą być wyrządzone po skutecznym ataku na uprzywilejowanego użytkownika lub komponent, ponieważ określone operacje będą wymagały udziału więcej niż tylko jednego użytkownika bądź komponentu.

Stosowanie zasad bezpieczeństwa w kontenerach

Jak zobaczysz dalej w tej książce, podział zadań w kontenerach może pomóc w zastosowaniu wszystkich wymienionych wcześniej zasad zapewnienia bezpieczeństwa.

Najmniejsze uprawnienia

Poszczególnym kontenerom można przydzielać odmienne zestawy uprawnień, każdy z nich będzie minimalnym zbiorem uprawnień niezbędnych do wykonania danego zadania.

Głęboka defensywa

Kontener stanowi kolejną granicę bezpieczeństwa, w ramach której można stosować linie obrony.

Ograniczenie płaszczyzny ataku

Podział monolitu na proste mikrousługi pozwala na opracowanie prostych łączących je interfejsów. Jeżeli się postarasz, zmniejszysz poziom skomplikowania rozwiązania i tym samym ograniczysz płaszczyznę ataku. Można się spotkać z kontrargumentem, że dodanie skomplikowanej warstwy orkiestracji do koordynacji kontenerów oznacza wprowadzenie kolejnej płaszczyzny ataku.

Ograniczenie pola rażenia

Jeżeli działająca w kontenerze aplikacja zostanie skutecznie zaatakowana, zastosowane mechanizmy obronne mogą uniemożliwić osobie atakującej wyjście poza dany kontener i uzyskanie dostępu do pozostałej części systemu.

Podział zadań

Uprawnienia i dane uwierzytelniające mogą być przekazywane tylko do wymagających ich kontenerów, więc złamanie jednego z takich zbiorów danych wrażliwych niekoniecznie będzie oznaczało ujawnienie wszystkich danych poufnych.

Wymienione zalety brzmią świetnie, choć w pewnym sensie mogą być uznane za czysto teoretyczne. W praktyce bardzo łatwo można je zniwelować poprzez kiepską konfigurację systemu, niewłaściwy sposób pracy z obrazami kontenerów lub też przez stosowanie niebezpiecznych praktyk. Gdy przeczytasz tę książkę, powinieneś mieć wystarczająco dobrą wiedzę pozwalającą na unikanie niebezpieczeństw, które wiążą się z wdrożeniami opartymi na kontenerach, a także będziesz potrafił wykorzystać wymienione wcześniej zalety.

Podsumowanie

W rozdziale ogólnie poznałeś rodzaje ataków, na jakie są narażone wdrożenia przeprowadzane na podstawie kontenerów. Zaprezentowałam również wprowadzenie do zasad bezpieczeństwa, które można stosować, aby bronić się przed wspomnianymi zagrożeniami. Dalej w książce zagłębisz się w mechanizmy kryjące się za kontenerami, a wtedy zrozumiesz, jak narzędzia zapewnienia bezpieczeństwa i najlepsze praktyki pozwalają na stosowanie tych zasad bezpieczeństwa.

System oparty na kontenerach. Jak dobrze jest zabezpieczony?

Imponująca możliwość skalowania oraz odporność na awarie skłania organizacje do uruchamiania swoich aplikacji w natywnych środowiskach chmury. Technologia kontenerów i orkiestracji stała się ostatnio bardzo modna. Jednak nawet tak nowoczesne systemy nie są wolne od zagrożeń. Aby zapewnić wdrożeniom kontenerowym najwyższy możliwy poziom bezpieczeństwa, trzeba zrozumieć mechanizmy działania kontenerów. Jako że powstają one przez połączenie różnych funkcji jądra systemu Linux, zapewnienie bezpieczeństwa kontenera oznacza zastosowanie wielu mechanizmów wykorzystywanych w komputerze gospodarza działającego pod kontrolą systemu operacyjnego Linux.

Książka jest przeznaczona dla programistów, menedżerów i specjalistów do spraw bezpieczeństwa odpowiedzialnych za systemy kontenerowe. Dzięki niej zrozumiesz, co się dzieje podczas uruchamiania aplikacji w kontenerach i jak działają różne mechanizmy zapewniania bezpieczeństwa. Przyswoisz kluczowe koncepcje, które ułatwią Ci ocenę ryzyka dla konkretnego systemu. Dowiesz się, jak w bezpieczny sposób tworzyć obrazy kontenerów, i zrozumiesz znaczenie poprawnej izolacji kontenerów. Zapoznasz się z podstawami korzystania z kluczy i certyfikatów służących do identyfikacji i nawiązywania bezpiecznych połączeń sieciowych między kontenerami. Nauczysz się korzystać z narzędzi do zapewniania bezpieczeństwa i unikania ataków. Dodatkowo zaprezentowany tu materiał został bogato zilustrowany gotowymi do przetestowania fragmentami kodu.

Najciekawsze zagadnienia ujęte w książce:

- mechanizmy ataków na wdrożenia oparte na kontenerach
- koncepcje systemu Linux istotne dla wdrożeń kontenerowych
- sposoby zabezpieczania kontenerów: najlepsze praktyki
- błędy w konfiguracji i luki w zabezpieczeniach kontenera
- bezpieczeństwo połączeń między kontenerami
- narzędzia do zapewnienia bezpieczeństwa

Liz Rice zajmuje jedno z kluczowych stanowisk w firmie Aqua Security. Specjalizuje się w systemach bezpieczeństwa kontenerów. Jest członkiem CNCF Technical Oversight Committee, współprowadziła konferencję KubeCon+ CloudNativeCon 2018 w Kopenhadze, Szanghaju i Seattle. Zdobyła imponujące doświadczenie w zespołowym tworzeniu oprogramowania oraz w pracy nad systemami rozproszonymi, a także w zakresie takich technologii jak VOD czy VoIP.

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

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-7228-3

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 59,00 zł