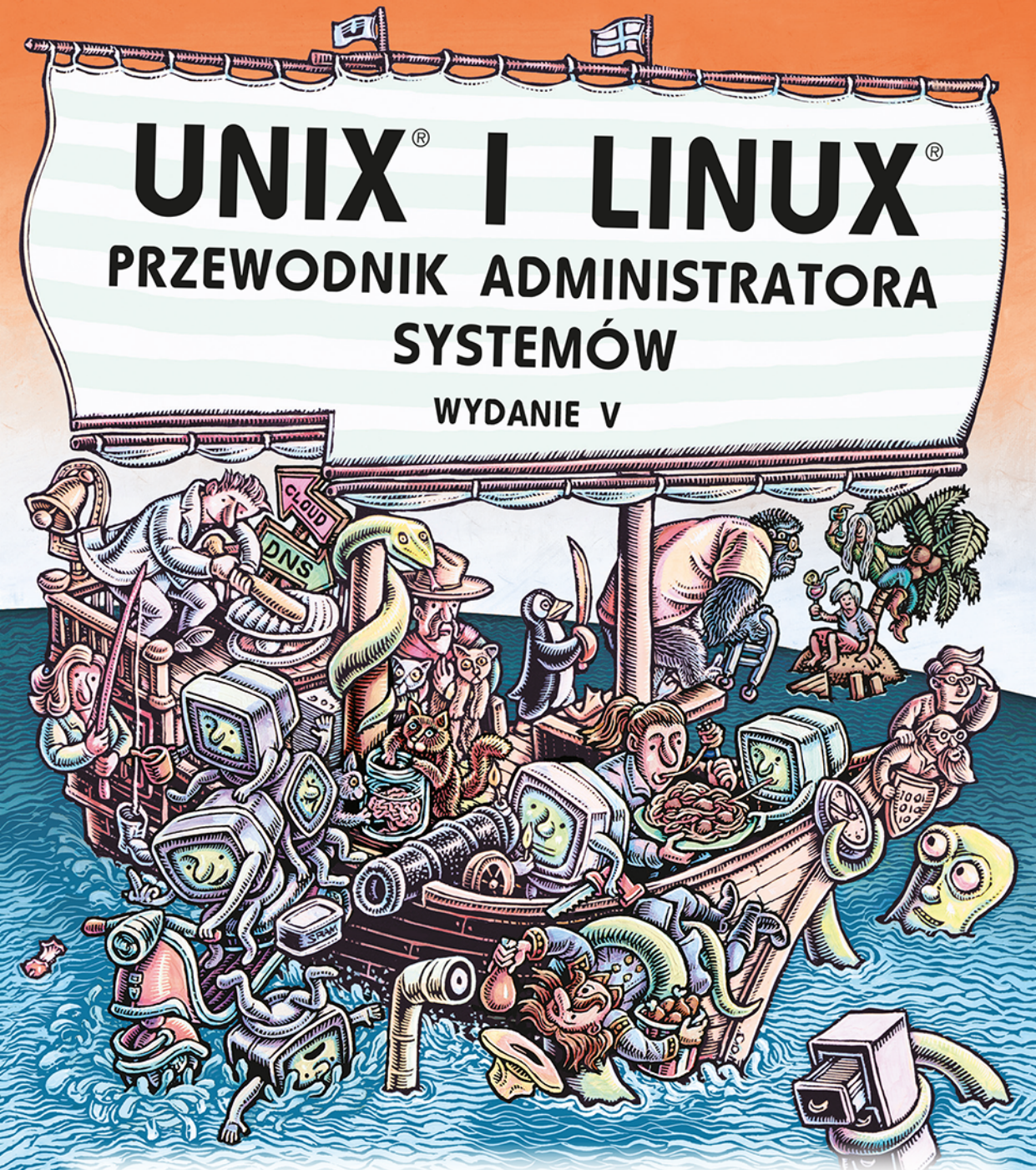
UNIX® I LINUX®

PRZEWODNIK ADMINISTRATORA SYSTEMÓW

WYDANIE V



EVI NEMETH • GARTH SNYDER • TRENT R. HEIN
BEN WHALEY • DAN MACKIN
James Garnett, Fabrizio Branca, Adrian Mouat

Helion

Tytuł oryginału: UNIX and Linux System Administration Handbook (5th Edition)

Tłumaczenie: Leszek Sagalara

ISBN: 978-83-8322-560-9

Authorized translation from the English language edition, entitled: UNIX AND LINUX SYSTEM ADMlNISTRATION HANDBOOK, Fifth Edition; ISBN 0134277554; by Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley, and Dan Mackin; published by Pearson Education, Inc, publishing as Addison-Wesley Professional. Copyright © 2018 Pearson Education, Inc.

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

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/unli5v
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

SPIS TREŚCI

2 Rozruch i demony zarządzania systemem 73

II. SIECI

III. PAMIĘĆ MASOWA

28 Monitoring 1069

28

MONITORING

Dążenie do monitorowania jest cechą wyróżniającą profesjonalnego administratora systemów. Niedoświadczeni administratorzy często pozostawiają systemy bez nadzoru i pozwalają na „wykrywanie" usterek, gdy sfrustrowany, rozgniewany użytkownik dzwoni do działu pomocy technicznej, ponieważ nie jest w stanie wykonać powierzonego mu zadania. Nieco bardziej uświadomione zespoły administracyjne tworzą platformę monitorowania, ale wyłączają powiadomienia po godzinach pracy, ponieważ są one zbyt uciążliwe. W obu przypadkach następuje walka z pożarem i ośmieszenie. Takie podejścia mają negatywny wpływ na przedsiębiorstwo, komplikują działania naprawcze i narażają administratorów na krytykę.

Profesjonalni administratorzy systemów przyjmują monitoring za swoją religię. Każdy system przed uruchomieniem jest dodawany do platformy monitorującej, a zestaw narzędzi kontrolnych jest regularnie testowany i dostrajany. Mierniki i trendy są oceniane proaktywnie, tak aby można było wykryć problemy, zanim dotkną one użytkowników lub wywołają zagrożenie bezpieczeństwa danych.

Duży serwis strumieniowania wideo on-line, o którym zapewne słyszałeś, ceni swój system telemetrii tak bardzo, że prędzej dopuszczono by tam do przestoju w świadczeniu usługi niż przestoju w monitorowaniu. Bez monitoringu i tak byłoby wiadomo, co się stało.

Filozofia „najpierw monitoring" (wraz z towarzyszącymi mu narzędziami) uczyni Cię superbohaterem wśród administratorów systemów. Dzięki niej lepiej poznasz swoje oprogramowanie i aplikacje, sprawniej rozwiążesz małe problemy, zanim przerodzą się w katastrofalne w skutkach awarie, staniesz się bardziej efektywny w znajdowaniu warunków wystąpienia błędów i usuwaniu problemów, a także zrozumiesz kwestię wydajności złożonych systemów. Monitorowanie poprawi również jakość Twojego życia, pozwalając na rozwiązanie większości problemów w dogodnym dla Ciebie czasie, a nie o trzeciej nad ranem w okresie świąt Bożego Narodzenia.

28.1. PRZEGLĄD MONITORINGU

Celem monitorowania jest zapewnienie funkcjonowania infrastruktury IT jako całości zgodnie z oczekiwaniami oraz zebranie, w przystępnej i łatwo przyswajalnej formie, danych przydatnych do zarządzania i planowania. Proste, prawda? Ten ogólny opis obejmuje jednak potencjalnie dość rozległy obszar.

Rzeczywiste systemy monitoringu różnią się w każdym możliwym wymiarze, ale wszystkie mają tę samą podstawową strukturę:

- z systemów i urządzeń będących przedmiotem zainteresowania pozyskiwane są surowe dane;
- platforma monitorująca dokonuje przeglądu danych i określa, jakie działania będą odpowiednie, zazwyczaj poprzez stosowanie administracyjnie określonych reguł;
- surowe dane i wszelkie decyzje podjęte przez system monitorowania przesyłane są na zaplecze, gdzie prowadzone są odpowiednie działania.

Rzeczywiste systemy monitoringu bywają zarówno trywialnie proste, jak i niezwykle skomplikowane. Przykładowo poniższy skrypt napisany w Perlu zawiera wszystkie wymienione wyżej elementy:

```
#!/usr/bin/env perl
$loadavg = (split /[\s,]+/, `uptime`)[10];
# Jeśli obciążenie jest większe niż 5, powiadom administratora
if ($loadavg > 5.0) {
  system 'mail -s "Obciążenie serwera jest za wysokie" dan@admin.com < /dev/null'
}
```

Skrypt ten uruchamia polecenie `uptime` w celu uzyskania średnich obciążeń systemu. Jeśli średnia wartość jednominutowego obciążenia jest większa niż `5.0`, skrypt wysyła wiadomość e-mail do administratora. Dane, ocena, reakcja.

Dawno, dawno temu „wyszukana" konfiguracja monitoringu składała się z zestawu takich skryptów, które były uruchamiane przez demona `cron` i nakazywały modemowi wysyłanie wiadomości na pagery administratorów. Obecnie na każdym etapie monitorowania potoku dostępnych jest wiele opcji.

Oczywiście nadal możesz pisać skrypty monitorujace i uruchamiać je przez program `cron`. Jeśli to naprawdę wszystko, czego potrzebujesz, za wszelką cenę staraj się zachować prostotę. Jeżeli jednak jesteś odpowiedzialny za więcej niż dwa serwery, takie doraźne podejście zwykle nie będzie wystarczające.

W dalszej części tego podrozdziału przyjrzymy się etapom potoku nieco dokładniej.

Instrumentacja

Szeroki zakres danych, które mogą się okazać użyteczne dla Twojej organizacji, obejmuje parametry wydajnościowe (czas odpowiedzi, wykorzystanie, szybkość transferu), parametry dotyczące dostępności (dostępność i nieprzerwany czas pracy), pojemność, zmiany stanu, wpisy dzienników, a nawet biznesowe wskaźniki efektywności, takie jak średnia wartość koszyka zakupów lub współczynnik konwersji kliknięć.

Ponieważ wszystko, co można zrobić na komputerze, jest potencjalnym przedmiotem monitorowania, systemy monitoringu są zazwyczaj agnostyczne, jeśli chodzi o źródła danych. Często są one dostarczane z wbudowanym wsparciem dla różnych elementów wejściowych. Nawet źródła danych, które nie są obsługiwane bezpośrednio, mogą być wprowadzane za pomocą kilku wierszy kodu dostosowawczego lub oddzielnej bramki danych, takiej jak StatsD (patrz podrozdział 28.4).

Przy tak dużej ilości danych, które aż się proszą o to, żeby je zbierać, trudną częścią projektowania systemu zbierania danych może być zdecydowanie o tym, co należy ignorować. Unikaj gromadzenia danych, które nie mają jasnego i możliwego do zrealizowania celu. Nadmiarowe zbieranie danych obciąża zarówno system monitoringu, jak i monitorowane jednostki. Prowadzi również do ukrywania wartości, które są naprawdę ważne, lecz giną w gąszczu informacji.

Niestety, często nie jest łatwo odróżnić użyteczne dane od śmieci. Musisz stale na nowo oceniać to, co jest monitorowane, i zastanawiać się nad tym, jak dane te będą się zachowywać przez cały okres życia systemu.

Rodzaje danych

Na najwyższym poziomie dane monitoringu można podzielić na trzy ogólne kategorie:

- **Wskaźniki czasu rzeczywistego**, charakteryzujące stan działania środowiska. Są to zazwyczaj liczby lub wartości logiczne. Ogólnie rzecz biorąc, obowiązkiem systemu monitorowania jest przetestowanie tych wskaźników w odniesieniu do oczekiwań i wygenerowanie alarmu w przypadku, gdy bieżąca wartość wykroczy poza określony wcześniej zakres lub próg.
- **Zdarzenia**, które często przybierają formę wpisów w plikach dziennika lub powiadomień typu „push" z podsystemów. Zdarzenia te, zwane czasami wskaźnikami opartymi na wzorcu, mogą sygnalizować zmianę stanu, stan alarmowy bądź inne działanie. Zdarzenia mogą być przetwarzane na dane numeryczne (np. sumę lub szybkość) lub bezpośrednio wywoływać odpowiedzi monitorujące[1].
- **Zagregowane i podsumowane trendy historyczne**, które są często zbiorami uszeregowanych czasowo wskaźników czasu rzeczywistego. Pozwalają one na analizę i wizualizację zmian w czasie.

[1] Wiele punktów danych gromadzonych przez oprogramowanie monitorujące aplikację należy do kategorii „zdarzenie", czasami dołączone są również dane ilościowe. Wzajemne powiązania pomiędzy zdarzeniami (np. „użytkownik przeglądał stronę *Ustawienia*, a następnie kliknął *Anuluj*, niczego nie zmieniając") są często pomocne w badaniu. Platformy monitorowania ogólnego przeznaczenia nie są zbyt dobre w przypadku tego rodzaju powiązań, co jest jednym z powodów, dla których monitorowanie aplikacji stało się całkiem osobną kategorią.

Pobieranie i przetwarzanie

Większość systemów monitoringu opiera się na centralnej platformie monitorującej, która absorbuje dane z monitorowanych systemów, przeprowadza odpowiednie przetwarzanie i stosuje reguły administracyjne w celu określenia, co powinno nastąpić w odpowiedzi.

Platformy pierwszej generacji, takie jak Nagios i Icinga, skupiały się na wykrywaniu pojawiających się problemów i reagowaniu na nie. Systemy te były rewolucyjne w swoich czasach i wprowadziły nas we współczesny świat monitoringu. Niemniej jednak z biegiem lat zostały one przyćmione przez rozwijającą się stopniowo w branży świadomość, że wszystkie monitorowane dane można uszeregować czasowo. Gdyby wartości nie różniły się od siebie, nie monitorowałbyś ich.

Oczywiste jest, że potrzebne było podejście bardziej ukierunkowane na dane. Jednak monitorowane dane są zazwyczaj tak obszerne, że nie można po prostu przenieść ich wszystkich do tradycyjnej bazy danych i pozwolić sobie na ich gromadzenie. Jest to recepta na niską wydajność i przepełnienie dysków.

Nowoczesne podejście polega na zorganizowaniu monitoringu wokół magazynu danych, który specjalizuje się w obsłudze danych uszeregowanych czasowo. Przez początkowy okres przechowywane są wszystkie dane, ale w miarę ich starzenia się magazyn stosuje coraz wyższe poziomy podsumowywania w celu ograniczenia wymagań odnośnie do pamięci masowej. Przykładowo magazyn może zachowywać dane z okresu godziny w rozdzielczości jednej sekundy, dane tygodniowe w rozdzielczości jednej minuty, a dane roczne w rozdzielczości godzinowej.

Dane historyczne są przydatne nie tylko do prezentacji na panelu, ale również jako punkt odniesienia dla porównań, np. czy obecny poziom błędów sieci przekracza średnią historyczną o 25% lub więcej.

Powiadomienia

Gdy masz już platformę monitorującą, zastanów się dokładnie, co zrobić z wynikami monitorowania. Priorytetem jest zazwyczaj poinformowanie administratorów i programistów o problemie, który wymaga uwagi.

Powiadomienia muszą się wiązać z podjęciem działań. Zorganizuj swój system monitorowania tak, aby każdy, kto otrzyma powiadomienie, musiał potencjalnie coś zrobić w odpowiedzi, nawet jeśli działanie jest czymś tak ogólnym jak „sprawdzić później, aby się upewnić, że to zostało zrobione". Powiadomienia, które są czysto informacyjne, wykształacają w personelu nawyk ich ignorowania.

W większości przypadków powiadomienia muszą wykraczać poza pocztę elektroniczną, aby były optymalnie skuteczne. W razie wystąpienia krytycznych problemów łatwym i wydajnym rozwiązaniem będą powiadomienia SMS (tj. wiadomości tekstowe) wysyłane na telefony komórkowe administratorów. Odbiorcy mogą ustawić dźwięki dzwonka i głośność telefonów tak, aby w razie potrzeby obudziły ich w środku nocy.

Powiadomienia powinny być również zintegrowane z implementacją ChatOps Twojego zespołu. Mniej krytyczne powiadomienia (takie jak statusy zadań, nieudane logowania i powiadomienia informacyjne) mogą być wysyłane do jednego lub kilku czatów, tak aby zainteresowane strony mogły otrzymywać podzbiory powiadomień, które mogłyby je interesować.

➲ *Więcej komentarzy na temat ChatOps można znaleźć w podrozdziale 31.1.*

Poza tymi podstawowymi kanałami możliwości powiadamiania są właściwie nieograniczone. Przykładowo system oświetlenia LED zmieniający kolory w zależności od stanu systemu może być przydatny do szybkiego wskazywania stanu w centrum danych lub sieciowym centrum operacyjnym. Inne opcje reagowania na sytuacje sygnalizowane przez systemy monitorowania obejmują m.in.:

- zautomatyzowane działania, takie jak zrzucanie baz danych lub rotacja dzienników;
- telefon do administratora;
- zamieszczenie danych na tablicy ściennej w celu ich upublicznienia;
- przechowywanie uszeregowanych czasowo danych w bazie do późniejszej analizy;
- nierobienie niczego i umożliwienie samemu systemowi dokonania późniejszego przeglądu.

Panele i interfejsy użytkownika

Oprócz ostrzegania o wyraźnie wyjątkowych okolicznościach jednym z głównych celów monitoringu jest przedstawienie stanu środowiska w sposób bardziej uporządkowany i łatwiejszy do przyswojenia niż szereg surowych danych. Takie wyświetlacze nazywane są ogólnie panelami (ang. *dashboards*).

Panele nawigacyjne są projektowane przez administratorów lub innych członków organizacji zainteresowanych konkretnymi aspektami środowiska. Panele wykorzystują kilka różnych technik przemiany surowych danych w infograficzne złoto.

Po pierwsze, są selektywne w tym, co prezentują. Koncentrują się na najważniejszych wskaźnikach dla danej dziedziny, ukazujących ogólny stan lub ogólną wydajność. Po drugie, dają wskazówki kontekstowe dotyczące znaczenia i importu prezentowanych danych. Przykładowo problematyczne liczby i stany są zazwyczaj wyświetlane na czerwono, a podstawowe wskaźniki są przedstawiane za pomocą czcionki o większych rozmiarach. Związki pomiędzy wartościami są prezentowane poprzez grupowanie. Po trzecie, pulpity wyświetlają serie danych jako wykresy, co ułatwia ich szybką ocenę.

Oczywiście, większość zebranych danych nigdy nie pojawia się na panelu. Dobrze jest też mieć system monitoringu, który posiada uniwersalny interfejs użytkownika, ułatwiający przeglądanie i modyfikowanie schematu danych, umożliwiający wykonywanie dowolnych zapytań bazodanowych oraz tworzenie wykresów dowolnie definiowanych sekwencji danych w locie.

28.2. KULTURA MONITORINGU

Ten rozdział dotyczy głównie narzędzi, ale kultura jest co najmniej równie ważna jak one. Wyruszając w podróż po świecie monitoringu, kieruj się poniższymi zasadami:

- Jeśli dany system lub dana usługa kogoś dotyczą lub ktoś jest od nich zależny, musi to być monitorowane. Koniec i kropka. Nic w środowisku, od którego zależy usługa lub użytkownik, nie może pozostać bez nadzoru.
- Jeżeli urządzenie produkcyjne, system lub usługa posiada cechy możliwe do monitorowania, powinny one być monitorowane. Nie pozwól, aby serwer z wymyślnym interfejsem zarządzania sprzętem spędził tygodnie na bezskutecznych próbach powiadomienia Cię o awarii wentylatora.
- Wszystkie konstrukcje o wysokiej dostępności muszą być monitorowane. Szkoda by się było dowiedzieć, że główny serwer nie działa, dopiero po tym, gdy awarii uległby także serwer rezerwowy.

- Monitorowanie nie jest opcjonalne. Każdy administaror systemów, deweloper, członek personelu operacyjnego, kierownik i kierownik projektu powinien w swoim planie pracy zarezerwować czas na monitorowanie.
- Dane z monitoringu (zwłaszcza dane historyczne) są przydatne dla wszystkich. Spraw, aby były one łatwo dostępne i widoczne, tak by każdy mógł z nich korzystać podczas analizy głównych przyczyn awarii, planowania, zarządzania cyklem życia systemów i wprowadzania ulepszeń architektonicznych. Nie szczędź swoich wysiłków ani zasobów na tworzenie i promowanie pulpitów monitorujących.
- Każdy powinien odpowiadać na ostrzeżenia. Monitorowanie nie jest problemem tylko pracowników operacyjnych. Wszystkie role techniczne powinny otrzymywać powiadomienia i współpracować przy rozwiązywaniu problemów. Takie podejście zachęca do rzetelnego analizowania ich przyczyn, niezależnie od tego, które osoby najbardziej nadają się do rozwiązania problemu.
- Poprawnie wdrożone monitorowanie wpływa pozytywnie na jakość życia administratora. Solidny system monitorowania uwalnia Cię od zamartwiania się o to, w jakim stanie są Twoje systemy, i daje innym możliwość wspierania Cię. Bez monitorowania i odpowiedniej dokumentacji będziesz musiał czuwać przy telefonie praktycznie 24 godziny na dobę przez 7 dni w tygodniu.
- Przeszkól osoby reagujące na sytuacje alarmowe, aby je naprawiały, a nie tylko łagodziły. Zidentyfikuj fałszywe lub zbyt natarczywe alarmy i dostosuj je tak, aby przestały się wyzwalać nieprawidłowo. Fałszywe alarmy zachęcają wszystkich do ignorowania systemu monitoringu.

28.3. PLATFORMY MONITORUJĄCE

Jeśli masz zamiar monitorować wiele systemów i więcej niż kilka wskaźników, warto poświęcić trochę czasu na wdrożenie pełnoprawnej platformy monitorującej. Jest to system ogólnego przeznaczenia, który zbiera dane z wielu źródeł, ułatwia wyświetlanie i podsumowywanie informacji o stanie oraz wprowadza ustandaryzowany sposób definiowania działań i ostrzeżeń.

Dobra wiadomość jest taka, że istnieje wiele możliwości do wyboru. Gorszą wiadomością jest to, że nie istnieje — jak na razie — jedna, doskonała platforma. Wybierając któreś z dostępnych rozwiązań, należy wziąć pod uwagę następujące kwestie:

- **Elastyczność gromadzenia danych.** Każda platforma może pobierać dane z różnych źródeł. Nie oznacza to jednak, że wszystkie platformy są pod tym względem równoważne. Pomyśl, z jakich źródeł danych rzeczywiście chcesz korzystać. Czy będziesz musiał odczytywać dane z bazy danych SQL? Z rekordów DNS? Z połączenia HTTP?
- **Jakość interfejsu użytkownika.** Wiele systemów oferuje konfigurowalne graficzne interfejsy użytkownika lub interfejsy WWW. Większość dobrze sprzedających się pakietów zachwala swoją zdolność wykorzystania szablonów JSON do prezentacji danych. Interfejs użytownika to nie tylko szum marketingowy; potrzebujesz interfejsu, który przekaże informacje w sposób jasny, prosty i zrozumiały. Może w Twojej organizacji potrzebujesz różnych interfejsów użytkownika dla różnych grup?

- **Koszt.** Niektóre komercyjne pakiety zarządzania są bardzo kosztowne. Wiele korporacji ceni sobie to, że ich witryna jest zarządzana przez wysokiej klasy system komercyjny. Jeśli nie jest to tak ważne dla Twojej organizacji, przyjrzyj się darmowym rozwiązaniom, takim jak Zabbix, Sensu, Cacti i Icinga.
- **Automatyczne wykrywanie.** Wiele systemów oferuje możliwość „wykrywania" Twojej sieci. Poprzez rozsyłanie poleceń `ping`, żądania SNMP, przeszukiwania tablic ARP i zapytania DNS identyfikują one wszystkie lokalne hosty i urządzenia. Wszystkie wdrożenia wykrywające, jakie widzieliśmy, działały całkiem dobrze, ale ich dokładność jest mniejsza w sieciach złożonych lub silnie zabezpieczonych zaporami.
- **Funkcje sprawozdawcze.** Wiele produktów może wysyłać powiadomienia e-mail, integrować się z ChatOps, wysyłać wiadomości tekstowe i automatycznie generować zgłoszenia dla popularnych systemów śledzenia błędów. Upewnij się, że wybrana przez Ciebie platforma umożliwia elastyczne raportowanie. Kto wie, z jakimi urządzeniami elektronicznymi będziesz miał do czynienia za kilka lat?

Platformy czasu rzeczywistego typu open source

Chociaż opisywane tu platformy — Nagios, Icinga i Sensu Core — robią wszystkiego po trochu, są znane z tego, że świetnie sobie radzą z pomiarami chwilowymi (lub opartymi na wskaźnikach).

Systemy te mają swoich zwolenników, ale jako narzędzia monitorowania pierwszej generacji stopniowo tracą przewagę nad systemami szeregów czasowych, które opisujemy w następnym punkcie. Większości ośrodków rozpoczynających od zera radzilibyśmy wybranie systemu opartego na szeregach czasowych.

Nagios i Icinga

Nagios i Icinga specjalizują się w powiadamianiu o błędach w czasie rzeczywistym. Chociaż systemy te nie pomagają w określeniu, o ile zwiększyło się wykorzystanie przepustowości łącza w ciągu ostatniego miesiąca, mogą jednak wykryć, że Twój serwer WWW przeszedł w tryb off-line.

Nagios i Icinga były pierwotnie odnogami jednego drzewa źródłowego, ale współczesna Icinga 2 została przepisana zupełnie na nowo. Pod wieloma względami pozostaje jednak kompatybilna z Nagiosem.

Oba systemy zawierają mnóstwo skryptów do monitorowania usług we wszystkich kształtach i rozmiarach oraz posiadają rozbudowane możliwości monitoringu SNMP. Prawdopodobnie ich największą zaletą jest modułowy i wysoce konfigurowalny system konfiguracji, który pozwala na pisanie własnych skryptów do monitorowania wszelkich możliwych wskaźników.

Jeśli czujesz się na siłach i masz skłonności masochistyczne, możesz napisać własne wtyczki monitorujące w Perlu, PHP, Pythonie czy nawet w C. Wiele standardowych metod powiadamiania jest wbudowanych: poczta elektroniczna, raporty WWW, wiadomości tekstowe itp. I podobnie jak w przypadku wtyczek monitorujących, możesz łatwo wprowadzić własne skrypty powiadomień i akcji.

Zarówno Nagios, jak i Icinga dobrze się sprawdzają w sieciach obsługujących mniej niż tysiąc hostów i urządzeń. Systemy te są łatwe do dostosowania i rozszerzania, a także zawierają potężne funkcje, takie jak nadmiarowość, zdalne monitorowanie i eskalacja powiadomień.

Jeśli instalujesz nową infrastrukturę monitoringu od podstaw, polecamy raczej Icingę 2 niż Nagiosa. Jej baza kodu jest na ogół czystsza i szybko zyskuje zwolenników oraz wsparcie społeczności. Z funkcjonalnego punktu widzenia interfejs użytkownika Icingi jest bardziej przejrzysty i szybszy, a także jest w stanie automatycznie budować zależności usług, co może mieć zasadnicze znaczenie w złożonych środowiskach.

Sensu

Sensu jest pełnowymiarową platformą monitorującą, dostępną zarówno w wersji open source (Sensu Core), jak i z płatnymi, komercyjnymi dodatkami. Posiada ultranowoczesny interfejs użytkownika i może uruchamiać wszelkie wtyczki monitorujące pochodzące z systemów Nagios, Icinga i Zabbix. Sensu zostało zaprojektowane jako zamiennik Nagiosa, więc kompatybilność z wtyczkami jest jedną z jego najbardziej atrakcyjnych cech. Sensu umożliwia łatwą integrację z powiadomieniami narzędzi Logstash i Slack, a jego instalacja jest wyjątkowo prosta.

Platformy szeregów czasowych typu open source

Wykrywanie bieżących problemów i reagowanie na nie to tylko jeden z aspektów monitoringu. Często równie ważna jest znajomość zmian wartości w czasie i ich związków z innymi wartościami. W celu zaspokojenia tej potrzeby powstały cztery popularne platformy szeregów czasowych: Graphite, Prometheus, InfluxDB i Munin.

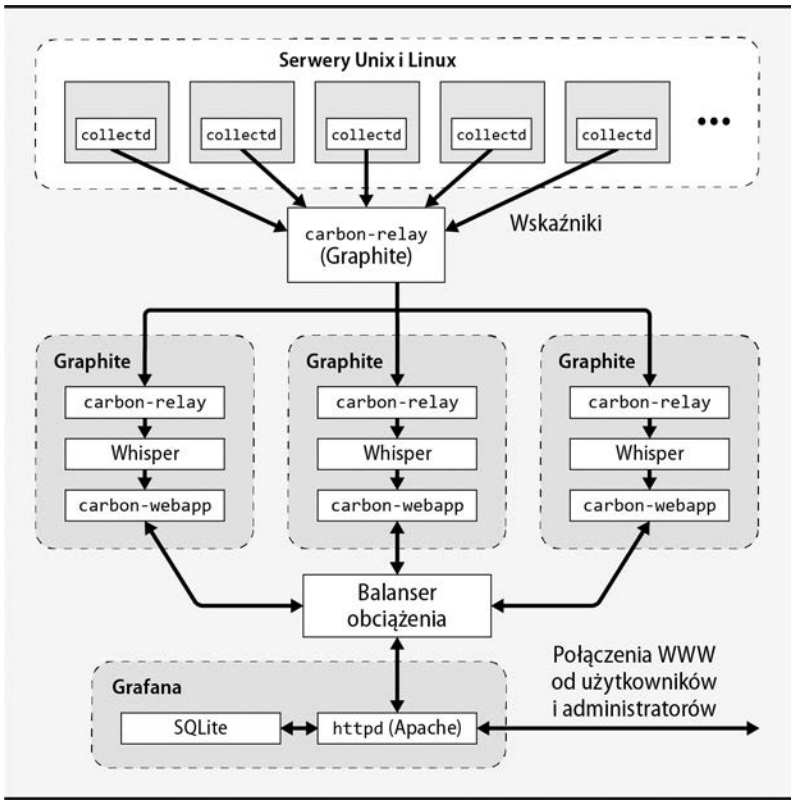
W tych systemach obliczem i sercem ekosystemu monitorowania jest baza danych. Różnią się one pod względem stopnia kompletności jako samodzielne systemy monitoringu i na ogół są zaprojektowane pod kątem świata bardziej modułowego niż ten, dla którego opracowano tradycyjne systemy, takie jak Icinga. Do zbudowania kompletnej platformy monitorowania konieczne może być dostarczenie dodatkowych komponentów.

Graphite

Graphite był niewątpliwie awangardą platform monitorowania szeregów czasowych nowej generacji. Jego trzonem jest elastyczna baza danych szeregów czasowych z łatwym w użyciu językiem zapytań. Przyczyną popularności hashtagu #monitoringlove i ogromnego wpływu, jaki Graphite wywarł na metody prezentacji danych w interfejsach użytkownika, jest sposób, w jaki agreguje on i podsumowuje wskaźniki. Rozpoczął on odchodzenie od monitorgu w przedziałach minutowych na rzecz skali mierzonej w ułamkach sekundy.

Jak można odgadnąć z nazwy, Graphite zawiera funkcje graficzne do wizualizacji WWW. Ten aspekt pakietu został jednak nieco przyćmiony przez podobną cechę Grafany. Graphite jest dziś lepiej znany z innych komponentów, takich jak Carbon i Whisper, które stanowią rdzeń systemu zarządzania danymi.

Platformę Graphite można łączyć z innymi narzędziami w celu stworzenia skalowalnego, rozproszonego, klastrowego środowiska monitorującego, które jest w stanie pochłaniać i raportować setki tysięcy wskaźników. Taki schemat architekturalny został przedstawiony na rysunku 28.1.



Rysunek 28.1. Klastrowa architektura platformy Graphite

Prometheus

Naszą ulubioną platformą szeregów czasowych jest obecnie Prometheus. Jest to wszechstronna platforma, która zawiera zintegrowane ze sobą elementy zbierania danych, analizy trendów i ostrzegania. Komponenty te są przyjazne zarówno dla administratorów, jak i deweloperów, co sprawia, że jest to doskonałe rozwiązanie dla DevOps. Nie pozwala jednak na klastrowanie, co może oznaczać, że nie jest ono odpowiednie dla ośrodków wymagających wysokiej dostępności.

InfluxDB

InfluxDB jest niezwykle przyjazną dla programistów platformą monitorowania szeregów czasowych, która obsługuje wiele języków programowania. Podobnie jak Graphite, InfluxDB to tak naprawdę tylko silnik bazodanowy szeregów czasowych. Będziesz musiał uzupełnić pakiet zewnętrznymi komponentami, takimi jak Grafana, aby stworzyć kompletny system monitoringu zawierający takie funkcje jak ostrzeganie.

Funkcje zarządzania danymi w InfluxDB są znacznie bardziej rozbudowane niż w przypadku rozwiązań wymienionych wyżej. Dodatkowe funkcje InfluxDB wprowadzają jednak pewną złożoność niepożądaną w typowych instalacjach.

InfluxDB ma dość kłopotliwą historię błędów i niezgodności. Jednak obecna wersja wydaje się stabilna i jest prawdopodobnie najlepszą alternatywą dla platformy Graphite, jeśli szukasz samodzielnego systemu zarządzania danymi.

Munin

Munin był niegdyś dość popularny, szczególnie w Skandynawii. Jest on zbudowany na bazie sprytnej architektury, w której wtyczki do zbierania danych nie tylko dostarczają dane, ale także informują system, w jaki sposób należy je przedstawić. Mimo że Munin wciąż doskonale nadaje się do użytku, w przypadku nowych wdrożeń należy rozważyć bardziej nowoczesne rozwiązania, takie jak Prometheus. W pewnych sytuacjach Munin jest jednak nadal użytecznym narzędziem do monitorowania aplikacji; patrz podrozdział 28.7.

Platformy open source do tworzenia wykresów

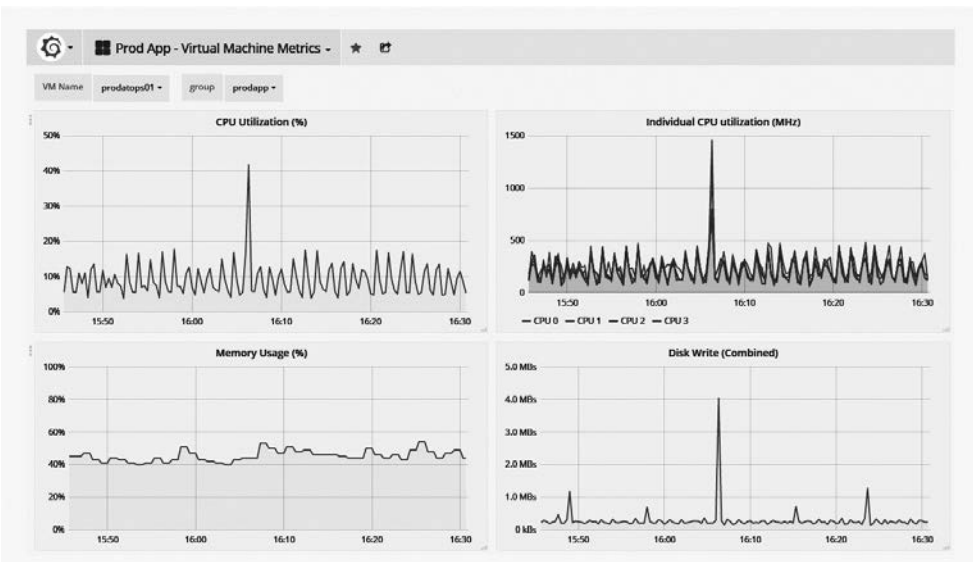
Dwa podstawowe sposoby tworzenia paneli i wykresów to funkcje graficzne wbudowane w platformę Graphite i nowszy pakiet, Grafana.

Graphite może przedstawiać graficznie dane pochodzące z innych magazynów niż Whisper (wbudowany komponent pamięci masowej pakietu Graphite), ale niekoniecznie jest to właściwy wybór.

Jako pakiet agnostyczny bazodanowo Grafana dość dobrze radzi sobie z obsługą obcych magazynów danych, w tym wszystkich wymienionych w poprzednim punkcie. Obsługuje ponad 30 baz źródeł. Pierwotnie Grafana miała być próbą poprawienia wykresów generowanych przez platformę Graphite, więc można z niej wygodnie korzystać również w tym środowisku.

Zarówno Grafite, jak i Grafana prezentują graficzny interfejs przypominający tablicę rozdzielczą, który może generować wizualizacje umożliwiające lepszy wgląd i łatwiejsze zarządzanie. Z ich pomocą można wyświetlać wszystko, począwszy od niskopoziomowych wskaźników systemowych, a skończywszy na wskaźnikach biznesowych. Testy porównawcze zazwyczaj wykazują przewagę Grafany ze względu na jej nieprzeciętny interfejs użytkownika i piękniejsze wykresy.

Na rysunku 28.2 przedstawiony został prosty pulpit Grafany.



Rysunek 28.2. Przykładowy pulpit Grafany

Komercyjne platformy monitorujące

Setki firm sprzedają oprogramowanie monitorujące, a co tydzień pojawiają się na rynku nowi konkurenci. Jeśli szukasz rozwiązania komercyjnego, powinieneś przynajmniej rozważyć opcje wymienione w tabeli 28.1.

Tabela 28.1. Popularne komercyjne platformy monitorujące

Platforma	Adres URL	Komentarz
Datadog	*datadoghq.com*	platforma monitorowania aplikacji w chmurze długa lista obsługiwanych systemów, aplikacji i usług
Librato	*librato.com*	obsługa istniejących wtyczek open source w trybie Plug and Play
Monitus	*monitus.net*	platforma monitorująca dla handlu elektronicznego
Pingdom	*pingdom.com*	platforma monitorująca oparta na SaaS[a]
SignalFx	*signalfx.com*	platforma SaaS z integracją z wieloma chmurami
SolarWinds	*solarwinds.com*	solidny monitoring sieciowy
Sysdig Cloud	*sysdig.com*	specjalność: monitorowanie i ostrzeganie w Dockerze łatwa korelacja zdarzeń pomiędzy usługami
Zenoss	*zenoss.com*	niezwykle złożona alternatywa dla systemu Icinga

[a] Nie wymaga instalacji oprogramowania. Dobre rozwiązanie tylko dla aplikacji WWW.

Większość firm nie powinna budować własnego systemu monitorowania, niezależnie od tego, czy miałby się on znajdować w chmurze, w hipernadzorcy centrum danych, czy w szafie. Outsourcing jest tańszy i bardziej niezawodny. Dlatego warto rozważyć Datadog, Librato, SignalFx lub Sysdig Cloud, jeśli potrzebujesz systemu monitorowania dla popularnego zestawu aplikacji bądź serwerów.

Analizując komercyjną platformę monitorującą, często najpierw zwracasz uwagę na jej cenę. Pamiętaj jednak o tym, aby zbadać również szczegóły jej działania:

- Czy da się ją łatwo zintegrować z Twoim systemem zarządzania konfiguracją?
- W jaki sposób system wdraża nowe wtyczki lub testy do Twoich hostów? Czy są one przesyłane, czy pobierane?
- Czy dobrze integruje się z istniejącą platformą powiadamiania, jeśli taką posiadasz?
- Czy Twoje środowisko dopuszcza połączenia zewnętrzne ułatwiające wprowadzenie rozwiązań monitorujących opartych na chmurze?

To tylko kilka z pytań, które powinieneś sobie zadać, analizując platformy. Ostatecznie najlepsza jest taka platforma, która ma prostą konfigurację, jest atrakcyjna cenowo i będzie z łatwością przyjęta przez Twoich użytkowników.

Hostowane platformy monitorujące

Jeśli nie jesteś zainteresowany konfigurowaniem i utrzymywaniem własnych narzędzi do monitorowania sieci, możesz rozważyć wybór platformy hostowanej (w chmurze). Istnieje wiele darmowych i komercyjnych rozwiązań, ale jednym z najpopularniejszych jest StatusCake (*statuscake.com*). Zewnętrzny dostawca ma ograniczone możliwości przeglądania wewnętrznych szczegółów Twojej sieci, ale rozwiązania hostowane dobrze się sprawdzają w zakresie walidacji stanu usług publicznych i witryn internetowych.

Dostawca hostowanej platformy monitorującej może również uwolnić Cię od ograniczeń związanych z funkcjonowaniem łącza internetowego Twojej organizacji. Jeśli do przesyłania powiadomień z wewnętrznego systemu monitorowania wykorzystujesz swoją sieć nadrzędną — co w końcu robi większość ośrodków — chciałbyś mieć pewność, że sama sieć nadrzędna będzie monitorowana i oprzyrządowana, aby w razie wystąpienia problemów można było zorganizować pracowników.

28.4. ZBIERANIE DANYCH

W poprzednim podrozdziale przyjrzeliśmy się różnym pakietom, które mogą pełnić funkcję centralnego mechanizmu monitorowania. Wybór i wdrożenie jednego z tych systemów jest jednak tylko pierwszą częścią procesu konfiguracji. Musisz się jeszcze upewnić, że interesujące Cię dane i zdarzenia trafią do centralnej platformy monitorującej.

Szczegóły procesu oprzyrządowania zależą od systemów, które chcesz monitorować, oraz od filozofii Twojej platformy monitorującej. W wielu przypadkach będziesz musiał napisać kilka prostych skryptów scalających, aby przekonwertować informacje o stanie na formę zrozumiałą dla platformy monitorującej. Niektóre platformy, takie jak Icinga, wyposażone są w zestaw wtyczek, które zbierają standardowe wskaźniki pomiarowe z powszechnie monitorowanych systemów. Inne platformy, takie jak Graphite i InfluxDB, w ogóle nie zapewniają mechanizmu wprowadzania danych i trzeba je uzupełnić o frontend, który wykona to zadanie.

W następnych punktach najpierw omówimy StatsD, frontend zbierania danych ogólnego przeznaczenia, a później przyjrzymy się pewnym narzędziom i technikom oprzyrządowania niektórych monitorowanych systemów.

StatsD — ogólny protokół przesyłania danych

StatsD został napisany przez inżynierów z serwisu Etsy jako sposób na śledzenie wszystkiego w ich własnym środowisku. Jest to oparty na UDP frontend pośredniczący, który zrzuca wszelkie przechodzące przez niego dane do platformy monitorującej w celu ich wykorzystania, przeliczenia i wyświetlenia. Supermocą StatsD jest jego zdolność do przyjmowania i dokonywania obliczeń na podstawie arbitralnych statystyk.

Demon StatsD firmy Etsy został napisany w Node.js. Jednak obecnie nazwa „StatsD" odnosi się bardziej do protokołu niż do jednego z wielu pakietów oprogramowania, które go implementują (prawdę mówiąc, nawet wersja z Etsy nie jest oryginalna; inspiracją do jej utworzenia był podobnie nazwany projekt firmy Flickr). Powstały wdrożenia napisane w wielu różnych językach, ale tutaj skupimy się na wydaniu Etsy.

StatsD zależy od Node.js, więc przed przystąpieniem do instalacji StatsD upewnij się, że ten pakiet został zainstalowany i odpowiednio skonfigurowany. Implementacja Etsy nie wchodzi w skład repozytoriów pakietów większości systemów operacyjnych, choć inne wersje StatsD są często dostępne; upewnij się, że ich nie pomylisz. Najprościej będzie sklonować wersję Etsy bezpośrednio z repozytorium GitHub:

```
$ git clone https://github.com/etsy/statsd
```

StatsD jest niewiarygodnie modułowy i może dostarczać przychodzące dane do różnych zapleczy i klientów. Spójrzmy na prosty przykład, który wykorzystuje platformę Graphite w roli zaplecza.

Aby się upewnić, że Graphite i StatsD komunikują się poprawnie, musisz zmodyfikować Carbon, składnik magazynujący Graphite. Dokonaj edycji pliku */etc/carbon/storage-schemas.conf* i dodaj fragment podobny do poniższego:

```
[stats]
pattern = ^stats.*
retentions = 10s:12h,1min:7d,10min:1y
```

Ta konfiguracja nakazuje, aby Carbon przez 12 godzin przechowywał dane z 10-sekundowych odstępów czasu. Carbon podsumowuje wygasające dane w odstępach jednominutowych i przechowuje to podsumowanie przez kolejne 7 dni. Dane o szczegółowości 10-minutowej przechowywane są przez cały rok. W dokonywaniu tych wyborów nie ma żadnej magii; musisz określić, co odpowiada potrzebom w zakresie przechowywania danych w Twojej organizacji i jakie dane mają być zbierane.

Dokładna definicja podsumowania danych szeregów czasowych różni się w zależności od rodzaju danych. Przykładowo, jeśli zliczasz błędy sieciowe, prawdopodobnie chciałbyś, aby podsumowanie dodawało wartości do siebie. Jeżeli szukasz mierników, które przedstawiają obciążenie lub stopień wykorzystania danych, prawdopodobnie będziesz potrzebował średniej. Być może będziesz musiał również określić odpowiednie sposoby postępowania z brakującymi danymi.

Zasady te są określone w pliku */etc/carbon/storage-aggregation.conf*. Jeśli nie masz jeszcze działającej instalacji Graphite, jego przykładowa konfiguracja będzie dla Ciebie dobrym punktem wyjścia:

```
/usr/share/doc/graphite-carbon/examples/storage-aggregation.conf.example
```

Poniżej przedstawiamy kilka sensownych ustawień domyślnych, które warto uwzględnić w pliku *storage-aggregation.conf*.

```
[min]
pattern = \.lower$
xFilesFactor = 0.1
aggregationMethod = min
[max]
pattern = \.upper(_\d+)?$
xFilesFactor = 0.1
aggregationMethod = max
[sum]
pattern = \.sum$
xFilesFactor = 0
aggregationMethod = sum
[count]
pattern = \.count$
xFilesFactor = 0
aggregationMethod = sum
[count_legacy]
pattern = ^stats_counts.*
xFilesFactor = 0
aggregationMethod = sum
[default_average]
pattern = .*
xFilesFactor = 0.3
aggregationMethod = average
```

Zauważ, że każdy blok konfiguracyjny posiada wzorzec wyrażenia regularnego, który próbuje dopasować nazwy serii danych. Bloki są odczytywane po kolei, a pierwszy dopasowany blok staje się specyfikacją kontrolną dla każdej serii danych. Przykładowo seria o nazwie *sample.count* odpowiada wzorcowi w bloku `[count]`. Wartości zostaną skumulowane poprzez zsumowanie punktów danych (`aggregationMethod = sum`).

Ustawienie `xFilesFactor` określa minimalną liczbę próbek potrzebnych do znaczącego zmniejszenia współczynnika próbkowania wskaźników. Jest ona wyrażona jako liczba z przedziału od 0 do 1 przedstawiająca wartość procentową wartości niezerowych, które muszą występować w warstwie o większej szczegółowości, aby warstwa sumowania nie miała wartości zerowej. Przykładowo ustawienie `xFilesFactor` w przedstawionych wyżej blokach `[min]` i `[max]` wynosi 10%, więc nawet pojedyncza wartość danych spełni to kryterium, biorąc pod uwagę nasze ustawienia w pliku *storage-schema.conf*. Wartość domyślna to 50%. Jeśli te ustawienia nie będą dobrze przemyślane, otrzymane dane będą niedokładne lub zostaną pominięte!

Możemy wysłać kilka danych testowych do StatsD za pomocą programu *netcat* (`nc`):

```
$ echo "sample.count:1|c" | nc -u -w0 statsd.admin.com 8125
```

Polecenie to przekazuje do zbioru danych *sample.count* wartość 1 jako licznik wskaźników (wskazany przez `c`). Pakiet przechodzi do portu 8125. w domenie *statsd.admin.com*; jest to port, na którym domyślnie nasłuchuje `statsd`. Jeśli ten pomiar pojawi się na panelu Graphite, będziesz gotowy do zbierania wszelkiego rodzaju monitorowanych danych za pośrednictwem jednego z wielu klientów StatsD. Listę klientów, które mogą się komunikować z programem StatsD, można znaleźć na jego stronie wiki w repozytorium GitHub (*github.com/etsy/statsd/wiki*). Możesz też napisać własnego klienta! Protokół jest prosty, a możliwości są nieograniczone.

Pozyskiwanie danych z wyjścia poleceń

Jeśli możesz coś prześledzić z wiersza poleceń, możesz to również prześledzić na swojej platformie monitorującej. Wszystko, czego potrzebujesz, to kilka wierszy skryptu scalającego do wyodrębnienia interesujących Cię danych, które następnie przekształcasz na format akceptowany przez platformę monitorującą.

Przykładowo polecenie `uptime` pokazuje czas pracy systemu, liczbę zalogowanych użytkowników i średnie obciążenie w ciągu ostatniej minuty, ostatnich 5 i 15 minut.

```
$ uptime
07:11:50 up 22 days, 10:13, 2 users, load average: 1.20, 1.41, 1.88
```

Człowiek jest w stanie błyskawicznie przetworzyć te dane wyjściowe i zobaczyć, że średnie obciążenie wynosi obecnie 1,20. Jeśli chcesz napisać skrypt, aby regularnie sprawdzać tę wartość lub wprowadzać ją do innego procesu monitorowania, możesz użyć poleceń do manipulacji tekstem w celu wyodrębnienia żądanej wartości:

```
$ uptime | perl -anF'[\s,]+' -e 'print $F[10]'[2]
1.20
```

Zastosowaliśmy tutaj język Perl do podziału wyjścia w każdym miejscu, gdzie występują spacje i przecinki, oraz do wyświetlenia zawartości dziesiątego pola (jest to średnia jednominutowego obciążenia). Voilà!

[2] To polecenie jest zależne od ustawień regionalnych, np. w polskiej wersji Linuksa nie zadziała, bo wartości dziesiętne są oddzielane przecinkiem, a nie kropką (1,20 zamiast 1.20) — *przyp. tłum.*

Chociaż w większości dziedzin Perl został przyćmiony przez nowocześniejsze języki, takie jak Python i Ruby, wciąż pozostaje królem szybkich manipulacji tekstem. Prawdopodobnie nie warto się uczyć Perla tylko w tym celu, ale umiejętność formułowania wyrafinowanych przekształceń tekstu jako jednowierszowych poleceń jest bardzo przydatna.

Możemy łatwo rozbudować to jednowierszowe polecenie do krótkiego skryptu, który określa wartość średniego obciążenia i przesyła ją do StatsD:

```
#!/usr/bin/env perl
use Net::Statsd;
use Sys::Hostname;
$Net::Statsd::HOST = 'statsd.admin.com';
$loadavg = (split /[\s,]+/, `uptime`)[10];
Net::Statsd::gauge(hostname . '.loadAverage' => $loadavg);
```

Porównaj ten skrypt z naszym jednowierszowym poleceniem wysyłającym dane testowe do StatsD z poprzedniego punktu i naszym jednowierszowym przetwarzaniem wyjścia `uptime`. W tym przypadku Perl musi uruchomić polecenie `uptime` i przetworzyć dane wyjściowe jako łańcuch, dlatego ta część wygląda nieco inaczej niż jej jednowierszowy odpowiednik (w którym wykorzystaliśmy tryb automatycznego podziału).

Zamiast używać `nc` do obsługi sieciowej transmisji danych do StatsD, używamy prostego skryptu osłonowego dla StatsD, który pobraliśmy z archiwum CPAN[3]. Jest to na ogół zalecane podejście; biblioteki są stabilniejsze niż doraźne rozwiązania i wyjaśniają przeznaczenie kodu.

Wiele poleceń może wygenerować więcej niż jeden format wyjściowy. Sprawdź stronę podręcznika danego polecenia, aby przejrzeć dostępne opcje, zanim spróbujesz przetworzyć jego wyniki. Niektóre formaty są znacznie łatwiejsze w obsłudze niż inne.

Kilka poleceń obsługuje format wyjściowy, który szczególnie ułatwia przetwarzanie. Inne posiadają konfigurowalne systemy wyjściowe; możesz w nich zażądać podania wartości tylko tych pól, które naprawdę są Ci potrzebne. Często polecenia mają opcję, która zapobiega wyświetlaniu opisowych wierszy nagłówka w wyjściu.

28.5. MONITOROWANIE SIECI

Wiele ośrodków zaczynało od wykorzystywania systemów monitorujących i paneli do monitorowania stanu sieci, więc jest to pierwszy z kilku rodzajów monitoringu, któremu przyjrzymy się nieco dokładniej. W kolejnych punktach zajmiemy się również monitoringiem systemów operacyjnych, aplikacji i usług oraz bezpieczeństwa.

Podstawową jednostką monitoringu sieciowego jest ping sieci, znany również jako pakiet *Echo Request* (żądanie echa) protokołu ICMP. Szczegóły techniczne omawiamy dokładniej w podrozdziale 13.12; opisujemy tam także polecenia `ping` i `ping6`, które inicjują wysyłanie sygnału `ping` z wiersza poleceń.

Koncepcja jest prosta: wysyłasz pakiet żądania echa do innego hosta w sieci, a jego implementacja IP w odpowiedzi zwraca pakiet do Ciebie. Jeśli otrzymasz odpowiedź na swój sygnał, wiesz, że wszystkie bramy sieciowe i urządzenia znajdujące się pomiędzy Tobą a docelowym hostem są sprawne. Wiesz również, że host docelowy jest włączony, a jego jądro jest uruchomione i działa. Ponieważ jednak sygnały `ping` obsługiwane są w stosie protokołu TCP/IP, nie dostarczają żadnych informacji na temat stanu oprogramowania wyższego poziomu, które może być uruchomione na docelowym komputerze.

[3] *Comprehensive Perl Archive Network*, *cpan.org*.

Sygnały `ping` nie narzucają zbyt dużego obciążenia sieci, więc można je wysyłać często — powiedzmy, że co dziesięć sekund. Zaprojektuj swoją strategię pingowania w sposób przemyślany, tak aby obejmowała wszystkie ważne bramy i sieci. Pamiętaj, że jeśli sygnał `ping` nie może przejść przez bramę, nie przejdą przez nią również dane monitoringu, które informują o awarii pingowania. Będziesz potrzebował co najmniej jednego zestawu pingów wychodzących z samego centralnego hosta monitorującego.

Bramy sieciowe nie muszą odpowiadać na pakiety `ping`, więc ich sygnał może zostać porzucony przez zajętą bramę. Nawet prawidłowo funkcjonująca sieć od czasu do czasu gubi pakiet, a więc nie uruchamiaj od razu alarmu przy pierwszej oznace błędów. Rozsądniejsze będzie zbieranie danych o sygnałach `ping` jako binarnych rekordów zdarzeń (przeszedł/nie przeszedł) i wprowadzanie ich do zagregowanych miar procentowych strat pakietów w dłuższym okresie.

Warto również dokonywać pomiarów przepustowości pomiędzy dwoma punktami w sieci. Można to zrobić za pomocą iPerf; szczegółowe informacje na ten temat można znaleźć w podrozdziale 13.13.

Większość urządzeń sieciowych obsługuje protokół SNMP(ang. *Simple Network Management Protocol*), będący branżowym standardem, jeśli chodzi o sposób nazywania i zbierania danych operacyjnych. Chociaż SNMP oddalił się znacznie od swoich sieciowych korzeni, uważamy, że jest on zbyt przestarzały, aby służyć do celów innych niż podstawowy monitoring sieci.

SNMP to dość obszerny temat, więc jego omówienie odłożymy na później (do podrozdziału 28.9, „Protokół SNMP”).

28.6. MONITOROWANIE SYSTEMÓW

Ponieważ jądro systemu steruje jego procesorem, pamięcią, systemami wejścia-wyjścia i urządzeniami, większość interesujących informacji o stanie systemu, które mógłbyś chcieć monitorować, znajduje się gdzieś wewnątrz jądra. Niezależnie od tego, czy analizujesz dany system ręcznie, czy też utworzyłeś zautomatyzowaną platformę monitorującą, potrzebujesz odpowiednich narzędzi do pozyskiwania i ujawniania informacji o stanie systemu. Większość jąder definiuje formalne kanały, którymi takie informacje są eksportowane.

Niestety, jądra są jak inne rodzaje oprogramowania; wyszukiwanie i usuwanie błędów oraz oprzyrządowanie to często coś, o czym myśli się po fakcie. Chociaż ostatnie lata przyniosły poprawę w zakresie przejrzystości, to zidetyfikowanie i zrozumienie dokładnego parametru, który chciałbyć monitorować, może być trudne, a czasami nawet niemożliwe.

Konkretną wartość można często uzyskać na kilka sposobów. Przykładowo w przypadku średnich obciążeń możesz odczytać wartości bezpośrednio z pliku */proc/loadavg* w systemach linuksowych lub za pomocą polecenia `sysctl -n vm.loadavg` w systemie FreeBSD. Średnie obciążenia są również zawarte w wyjściach poleceń `uptime`, `w`, `sar` i `top` (choć `top` jest kiepskim wyborem do użytku nieinteraktywnego). Ogólnie rzecz biorąc, najłatwiejsze i najskuteczniejsze jest uzyskanie dostępu do wartości bezpośrednio z jądra (poprzez `sysctl` lub */proc*), jeśli jest to możliwe.

➲ *Więcej informacji na temat systemu plików* /proc *można znaleźć w podrozdziale 11.4.*

Platformy monitorujące, takie jak Nagios i Icinga, zawierają bogaty zestaw wtyczek monitorujących rozwijanych przez społeczność, które można wykorzystać, aby uzyskać dostęp do powszechnie monitorowanych elementów. Często są to również zwykłe skrypty, które uruchamiają polecenia i przetwarzają dane wyjściowe, ale są już przetestowane i oczyszczone z błędów i działają na wielu platformach. Jeśli nie możesz znaleźć wtyczki, która zapewni Ci interesującą Cię wartość, możesz napisać własną.

Polecenia dla systemów monitorowania

Tabela 28.2 zawiera listę poleceń, które są powszechnie używane w monitoringu. Wiele z nich daje bardzo różne wyniki w zależności od podanych opcji wiersza poleceń, więc sprawdź szczegóły na stronach podręcznika systemowego.

Tabela 28.2. Polecenia zwracające powszechnie monitorowane parametry

Polecenie	Dostępne informacje
`df`	wolne i zajęte miejsce na dysku oraz i-węzłach
`du`	rozmiary katalogów
`free`	ilość pamięci wolnej, zajętej i wirtualnej (wymiany)
`iostat`	wydajność i przepustowość dysku
`lsof`	otwarte pliki i porty sieciowe
`mpstat`	wykorzystanie poszczególnych procesorów w systemach wieloprocesorowych
`vmstat`	statystyki dotyczące procesów, procesorów i pamięci

Polecenie `sar` (skrót od *system activity report*, raport o aktywności systemu) jest swoistym scyzorykiem szwajcarskim, jeśli chodzi o ekstrakcję danych z wiersza poleceń. Polecenie to ma skomplikowaną historię; pierwotnie zostało wprowadzone do uniksowego System V w latach osiemdziesiątych ubiegłego wieku[4]. Jego główną zaletą jest to, że zostało ono zaimplementowane na wielu różnych systemach, co zwiększa przenośność zarówno dla skryptów, jak i dla administratorów systemów. Niestety, port BSD nie jest już utrzymywany.

Poniższy przykład żąda raportowania co dwie sekundy przez okres jednej minuty (tj. 30 raportów). Argument `DEV` jest słowem kluczowym, a nie zmienną dla nazwy urządzenia lub interfejsu.

```
$ sar -n DEV 2 30
17:50:43  IFACE  rxpck/s  txpck/s  rxbyt/s  txbyt/s  rxcmp/s  txcmp/s  rxmcst/s
17:50:45  lo       3.61     3.61   263.40   263.40     0.00     0.00      0.00
17:50:45  eth0    18.56    11.86  1364.43  1494.33     0.00     0.00      0.52
17:50:45  eth1     0.00     0.00     0.00     0.00     0.00     0.00      0.00
```

Przykład ten pochodzi z komputera z systemem Linux z dwoma interfejsami sieciowymi. Wyjście zawiera zarówno chwilowe, jak i średnie odczyty wykorzystania interfejsu, przy czym każdy odczyt wyrażony jest zarówno w bajtach, jak i w pakietach. Drugi interfejs (*eth1*) najwyraźniej nie jest używany.

[4] Administratorów starej daty można rozpoznać po ich biegłości w posługiwaniu się poleceniem `sar`.

collectd — pozyskiwanie ogólnych danych systemowych

W miarę jak administrowanie systemami ewoluowało od zmagania się z pojedynczymi systemami do zarządzania flotą wirtualnych instancji, użytkowanie prostych narzędzi wiersza poleceń zaczęło stwarzać wiele problemów w dziedzinie monitorowania. Choć pisanie skryptów do zbierania i analizowania parametrów jest utylitarnym i elastycznym podejściem, utrzymanie spójności tej bazy kodów w wielu systemach szybko staje się kłopotliwe. Nowoczesne narzędzia, takie jak `collectd`, `sysdig` i `dtrace`, oferują bardziej skalowalne podejście do zbierania tego typu danych.

Gromadzenie statystyk systemowych powinno być procesem ciągłym, a uniksowe rozwiązanie dla trwającego zadania polega na utworzeniu demona, który się tym zajmie. Jest nim `collectd`, demon zbierania statystyk systemowych.

To popularne i dojrzałe narzędzie dla systemów Linux i FreeBSD. Zazwyczaj `collectd` działa w systemie lokalnym, zbiera dane pomiarowe w określonych odstępach czasu i zapisuje wartości wynikowe. Możesz również skonfigurować `collectd` do pracy w trybie klient-serwer, w którym `collectd` (lub kilka jego instancji) agreguje dane z grupy innych serwerów.

Specyfikacja parametrów, które mają być zbierane, oraz miejsc docelowych, do których są one zapisywane, jest elastyczna; dostępnych jest ponad 100 wtyczek, które można dostosować do własnych potrzeb. Po uruchomieniu `collectd` może być odpytywany przez platformę monitorującą (taką jak Icinga lub Nagios) w celu monitorowania na bieżąco albo może przekazywać dane do takich platform jak Graphite bądź InfluxDB w celu analizy szeregów czasowych.

Oto przykładowy plik konfiguracyjny `collectd`:

```
## /etc/collectd/collectd.conf
Hostname client1.admin.com
FQDNLookup false
Interval 30
LoadPlugin syslog
<Plugin syslog>
    LogLevel info
</Plugin>
LoadPlugin cpu
LoadPlugin df
LoadPlugin disk
LoadPlugin interface
LoadPlugin load
LoadPlugin memory
LoadPlugin processes
LoadPlugin rrdtool
<Plugin rrdtool>
    DataDir "/var/lib/collectd/rrd"
</Plugin>
```

Ta podstawowa konfiguracja zbiera wiele interesujących statystyk systemowych co 30 sekund i zapisuje pliki danych zgodne z RRD (ang. *Round Robin Database*, cykliczna baza danych) w katalogu */var/lib/collectd/rrd*.

sysdig i dtrace — śledzenie działań w systemie

`sysdig` (Linux) i `dtrace` (BSD) kompleksowo instrumentują aktywność zarówno jądra, jak i procesów użytkownika. Obejmują one komponenty, które są wstawiane do samego jądra, ujawniając nie tylko głęboko ukryte parametry jądra, ale także wywołania systemowe i inne statystyki wydajności. Narzędzia te są czasami określane jako „Wireshark dla jądra i procesów".

➲ *Więcej informacji na temat Wiresharka można znaleźć w podrozdziale 13.12.*

Oba te narzędzia są skomplikowane. Warto jednak podjąć wyzwanie i zapoznać się z nimi. Weekend spędzony na nauce jedego z nich da Ci niesamowite nowe supermoce i zapewni Ci status gościa z listy VIP-ów na spotkaniu koktajlowym dla administratorów systemów.

`sysdig` potrafi obsługiwać kontenery, dzięki czemu zapewnia niezwykłą widoczność w środowiskach, w których używane są narzędzia takie jak Docker i LXC. `sysdig` jest dystrybuowany jako oprogramowanie open source i można go zintegrować z innymi narzędziami monitoringu, takimi jak Nagios lub Icinga. Deweloperzy oferują również komercyjną usługę monitoringu (Sysdig Cloud), która posiada pełne możliwości monitorowania i ostrzegania.

➲ *Więcej informacji na temat Dockera i kontenerów można znaleźć w rozdziale 25.*

28.7. MONITOROWANIE APLIKACJI

Na szczycie piramidy oprogramowania znajduje się Święty Graal: monitoring aplikacji. Ten rodzaj monitorowania jest dość niejasno zdefiniowany, ale ogólna koncepcja polega na tym, że próbuje się sprawdzać status i wydajność poszczególnych elementów oprogramowania, a nie systemów czy sieci jako całości. W wielu przypadkach monitorowanie aplikacji może sięgać do tych systemów i profilować ich wewnętrzne działania.

Aby mieć pewność, że monitorujesz właściwe rzeczy, musisz włączyć do pracy jednostki biznesowe i programistów, by opowiedzieli Ci więcej o swoich zainteresowaniach i obawach. Jeśli masz witrynę internetową działającą np. na zestawie LAMP, prawdopodobnie będziesz chciał monitorować czasy ładowania stron, oznaczać krytyczne błędy PHP, trzymać zakładki w bazie danych MySQL i monitorować konkretne problemy, takie jak nadmierne próby połączeń.

Monitorowanie tej warstwy może być złożone, ale jest to również obszar, w którym monitorowanie staje się atrakcyjne. Wyobraź sobie, że możesz monitorować (i prezentować na swoim pięknym panelu Grafany) liczbę widżetów, które sprzedałeś w ciągu minionej godziny, lub średni czas, przez jaki dany przedmiot pozostaje w koszyku. Jeśli pokażesz twórcom aplikacji i właścicielom procesów taki poziom funkcjonalności, zazwyczaj otrzymasz natychmiastowe wsparcie w celu dodania dodatkowego monitorowania, a może nawet pomoc w jego wdrożeniu. W końcu ta warstwa monitoringu stanie się nieoceniona dla biznesu, a Ty zaczniesz być postrzegany jako mistrz monitoringu, wskaźników i analizy danych.

Monitorowanie na poziomie aplikacji pozwala uzyskać dodatkowy wgląd w inne zdarzenia w środowisku. Przykładowo, jeśli sprzedaż widżetów szybko spada, może to być oznaką tego, że jedna z sieci reklamowych nie działa.

Monitorowanie dzienników

W swojej najbardziej podstawowej formie monitorowanie dzienników polega na przeszukiwaniu plików dzienników w celu zdobycia interesujących danych, które chciałbyś monitorować, wyciąganiu tych danych i przetwarzaniu ich do postaci zdatnej do analizy, wyświetlania i ostrzegania. Ponieważ komunikaty dziennika składają się z tekstu o dowolnej postaci, implementacja tego potoku może mieć bardzo różny poziom złożoności; może to być zarówno trywialne, jak i mocno skomplikowane.

Zazwyczaj dziennikami najlepiej zarządza się za pomocą kompleksowego systemu agregacji zaprojektowanego specjalnie w tym celu. Omawiamy takie systemy w podrozdziale 10.4,

„Zarządzanie dziennikami na dużą skalę". Chociaż systemy te koncentrują się przede wszystkim na centralizacji danych dzienników oraz ułatwianiu ich wyszukiwania i przeglądania, większość systemów agregacji obsługuje również wartości progowe, alarmy i funkcje raportowania.

Jeśli potrzebujesz automatyzacji przeglądania dziennika dla kilku konkretnych celów i nie masz ochoty inwestować w bardziej ogólny system zarządzania dziennikami, zalecamy skorzystanie z pary narzędzi działających w mniejszej skali: `logwatch` i OSSEC.

`logwatch` jest elastycznym analizatorem dzienników, zorientowanym na przetwarzanie wsadowe. Jego podstawowym zastosowaniem jest tworzenie dziennych podsumowań zdarzeń raportowanych w dziennikach. Możesz uruchamiać `logwatch` częściej niż raz dziennie, ale nie został on zaprojektowany do monitorowania w czasie rzeczywistym. Do tego warto wykorzystać OSSEC, który omawiamy w podrozdziale 27.5. OSSEC jest promowany jako narzędzie bezpieczeństwa, ale jego architektura jest na tyle ogólna, że jest również przydatny dla innych rodzajów monitoringu.

Supervisor + Munin — proste rozwiązanie dla ograniczonych zastosowań

Wszechstronna, kompleksowa platforma, taka jak Icinga czy Prometheus, może się okazać nieodpowiednia dla Twoich potrzeb lub Twojego środowiska. Co zrobić, jeśli jesteś zainteresowany tylko monitorowaniem jednego konkretnego procesu aplikacji i nie chcesz zawracać sobie głowy pełnoprawną platformą monitorującą? Rozważ połączenie ze sobą narzędzi Munin i Supervisor. Są one łatwe w instalacji, wymagają niewielu czynności konfiguracyjnych i dobrze ze sobą współpracują.

Supervisor i jego proces serwera `supervisord` pomagają monitorować procesy i generować zdarzenia lub powiadomienia, gdy procesy kończą działanie albo zgłaszają wyjątek. System ten przypomina w swojej istocie Upstart lub te części `systemd`, które odpowiadają za zarządzanie procesami.

Jak już wspominaliśmy, Munin jest platformą monitorującą ogólnego przeznaczenia, ze szczególnym uwzględnieniem monitorowania aplikacji. Jest napisany w języku Perl i wymaga, aby na każdym systemie, który chcesz monitorować, uruchomiony był agent zwany węzłem (ang. *Munin Node*). Konfiguracja nowego węzła jest prosta: zainstaluj pakiet `munin-node`, zmodyfikuj plik *munin-node.conf*, aby wskazać komputer nadrzędny, i gotowe.

Przedstawiając zgromadzone dane, Munin domyślnie tworzy wykresy za pomocą silnika opartego na RRDtool, więc jest to przyjemny i wymagający niewielu czynności konfiguracyjnych sposób na uzyskanie graficznej informacji zwrotnej. Munin posiada ponad 300 wtyczek oraz prawie 200 udostępnionych bibliotek. Prawdopodobnie znajdziesz już istniejącą wtyczkę, która zaspokoi Twoje potrzeby. Jeśli nie, łatwo jest napisać nowy skrypt do wykonania przez `munin-node`.

Komercyjne narzędzia do monitorowania aplikacji

Jeśli wpiszesz w Google hasło „narzędzie do monitorowania aplikacji", odkryjesz wiele stron przedstawiających komercyjne produkty do oceny. Aby dochować należytej staranności, powinieneś również prześledzić ostatnie dyskusje na temat monitorowania wydajności aplikacji (APM, ang. *application performance monitoring*).

Znajdziesz tam wiele odniesień do DevOps, z uzasadnionych powodów: monitorowanie aplikacji i APM to kluczowe elementy DevOps. Dostarczają one danych ilościowych; zespoły mogą ich użyć, aby zdecydować, którym obszarom działania na rzecz poprawy wydajności i stabilności przyniosą największe korzyści.

➲ *Więcej informacji na temat DevOps można znaleźć w podrozdziale 31.1.*

Uważamy, że w tym zakresie wyróżniają się New Relic (*newrelic.com*) oraz AppDynamics (*appdynamics.com*). Możliwości tych systemów pod wieloma względami nakładają się na siebie, ale AppDynamics zazwyczaj koncentruje się na rozwiązaniach monitorujących wykorzystujących „pełny zestaw", podczas gdy New Relic zajmuje się bardziej profilowaniem zachowania w obrębie samej warstwy aplikacji.

Niezależnie od tego, w jaki sposób monitorujesz swoje aplikacje, ważne jest, aby w ten proces byli zaangażowani programiści. Pomogą oni zapewnić monitorowanie wszystkich ważnych wskaźników. Ścisła współpraca w zakresie monitorowania sprzyja rozwijaniu wzajemnych relacji pomiędzy zespołami i ogranicza powielanie działań.

28.8. MONITOROWANIE BEZPIECZEŃSTWA

Monitoring bezpieczeństwa to osobny wszechświat. Ten obszar praktyki operacyjnej nazywany jest czasami operacjami bezpieczeństwa, w skrócie SecOps.

Można wymienić dziesiątki narzędzi i usług, zarówno open source, jak i komercyjnych, które pomagają w monitorowaniu bezpieczeństwa środowiska. Podmioty zewnętrzne, zwane dostawcami zarządzanych usług bezpieczeństwa (MSSP, ang. *managed security service providers*), świadczą usługi na zasadzie outsourcingu[5]. Mimo istnienia wszystkich tych możliwości naruszenia bezpieczeństwa są nadal powszechne i często pozostają niewykryte przez miesiące lub lata.

Prawdopodobnie najważniejszą rzeczą, jaką należy wiedzieć o monitorowaniu bezpieczeństwa, jest to, że nie wystarczy tu zautomatyzowane narzędzie lub zautomatyzowana usługa. Musisz wdrożyć kompleksowy program bezpieczeństwa, zawierający standardy dotyczące zachowań użytkowników, przechowywania danych i procedur reagowania na incydenty, by wymienić tylko kilka elementów. Podstawy tego zagadnienia omawiamy w rozdziale 27., „Bezpieczeństwo".

Z automatyczną, ciągłą strategią monitorowania należy zintegrować dwie podstawowe funkcje bezpieczeństwa: weryfikację integralności systemu i wykrywanie włamań.

Weryfikowanie integralności systemu

Weryfikacja integralności systemu, często nazywana monitoringiem integralności plików (FIM, *file integrity monitoring*), jest walidacją bieżącego stanu systemu w stosunku do znanego stanu wyjściowego. Najczęściej walidacja ta porównuje zawartość plików systemowych (jądro, polecenia wykonywalne, pliki konfiguracyjne) z kryptograficzną sumą kontrolną, taką

[5] Outsourcing operacji bezpieczeństwa zawsze wydaje się atrakcyjny; zapewnienie bezpieczeństwa Twojemu środowisku staje się wtedy problemem kogoś innego. Ale pomyśl o tym tak: czy czułbyś się komfortowo, płacąc komuś za pilnowanie Twojego wypełnionego gotówką portfela, leżącego wraz z 10 000 innych portfeli na stole na zatłoczonym dworcu kolejowym? Jeśli tak, to MSSP może być dla Ciebie dobrym rozwiązaniem!

jak SHA-512[6]. Jeżeli suma kontrolna pliku w uruchomionym systemie różni się od sumy kontrolnej wersji wyjściowej, administrator zostanie o tym powiadomiony. Oczywiście, należy brać pod uwagę regularne działania konserwacyjne, takie jak planowane zmiany, aktualizacje i poprawki; nie wszystkie zmiany są podejrzane.

Najczęściej stosowanymi platformami FIM są Tripwire i OSSEC, opisane bardziej szczegółowo w podrozdziale 27.5. Linuksowa wersja AIDE również zawiera monitorowanie integralności plików, ale wersja FreeBSD niestety nie ma tego komponentu.

Prostsze jest często lepsze. Wspaniałą standardową opcją FIM jest program `mtree`, który wywodzi się z FreeBSD, a niedawno został przeniesiony do Linuksa. `mtree` to prosty sposób na monitorowanie stanu plików i zmian ich zawartości, łatwy do zintegrowania ze skryptami monitorującymi. Oto przykład szybkiego skryptu wykorzystującego `mtree`:

```
#!/bin/bash
if [ $# -eq 0 ]; then
    echo "mtree-check.sh [-bv]"
    echo "-b = utworzenie stanu wyjściowego"
    echo "-v = weryfikacja względem stanu wyjściowego"
    exit
fi
## ziarno
KEY=93948764681464
## katalog bazowy
DIR=/usr/local/lib/mtree-check
if [ $1 = "-b" ]; then
    rm -rf $DIR/mtree_*
    cd $DIR
    mtree-c -K sha512 -s $KEY -p /sbin > mtree_sbin
fi
if [ $1 = "-v" ]; then
    cd $DIR
    mtree -s $KEY -p /sbin < mtree_sbin | \
        mail -s "mtree: test integralności `hostname`" dan@admin.com
fi
```

Po dodaniu opcji `-b` skrypt tworzy i zapisuje stan wyjściowy. Po ponownym uruchomieniu z opcją `-v` sprawdza bieżącą zawartość katalogu */sbin* względem stanu wyjściowego.

Podobnie jak w przypadku wielu aspektów zarządzania systemem, stworzenie platformy FIM i jej obsługa w czasie to dwie zupełnie różne sprawy. Do przechowywania danych i odpowiadania na powiadomienia FIM potrzebny jest zdefiniowany proces. Sugerujemy wprowadzanie informacji z platformy FIM do infrastruktury monitorowania i ostrzegania, tak aby nie były one odsuwane na bok lub ignorowane.

Monitorowanie wykrywania włamań

Powszechnie stosuje się dwa rodzaje systemów wykrywania włamań (IDS, ang. *intrusion detection systems*): hostowe (HIDS, ang. *host-based IDS*) i sieciowe (NIDS, ang. *network-based IDS*). Systemy NIDS badają ruch przechodzący przez sieć i próbują zidentyfikować niespodziewane lub podejrzane wzorce. Najpopularniejszy system NIDS jest oparty na Snort; omawiamy go szczegółowo w podrozdziale 27.5.

[6] Akceptowalne algorytmy skrótu zmieniają się w czasie. Na przykład algorytm MD5 nie jest już uważany za kryptograficznie bezpieczny i nie powininno się go używać.

Systemy HIDS działają jako zestaw procesów na każdym systemie. Zazwyczaj śledzą one różne rzeczy, w tym połączenia sieciowe, czasy modyfikacji plików i sumy kontrolne, dzienniki demonów i aplikacji, wykorzystanie podwyższonych uprawnień oraz inne wskazówki, które mogą sygnalizować działanie narzędzi zaprojektowanych w celu ułatwienia nieautoryzowanego dostępu (programów typu „root kit"). HIDS nie jest kompleksowym rozwiązaniem dla bezpieczeństwa, ale stanowi cenny element kompleksowego podejścia.

Dwie najpopularniejsze platformy HIDS typu open source to OSSEC (ang. *Open Source SECurity*) oraz AIDE (ang. *Advanced Intrusion Detection Environment*). Z naszego doświadczenia wynika, że OSSEC jest najlepszym wyborem. Chociaż AIDE jest świetną platformą FIM dla systemu Linux, OSSEC posiada bogatszy zestaw funkcji. Może być nawet używany w trybie klient-serwer, który obsługuje klientów innych niż Unix, takich jak Microsoft Windows, i wiele urządzeń infrastruktury sieciowej.

Podobnie jak alarmy FIM, dane HIDS są użyteczne tylko wtedy, gdy poświęcamy tej platformie należytą uwagę. HIDS nie jest podsystemem typu „ustaw i zapomnij"; będziesz musiał zintegrować alarmy HIDS z Twoim ogólnym systemem monitorowania. Najskuteczniejszą strategią, jaką udało nam się znaleźć do rozwiązania tego problemu, jest automatyczne otwieranie zgłoszeń na ostrzeżenia HIDS w Twoim systemie zgłoszeniowym. Następnie można dodać kontrolę monitoringu, która ostrzeże o wszystkich nierozwiązanych zgłoszeniach HIDS.

28.9. PROTOKÓŁ SNMP

Przed laty branża sieciowa zdecydowała, że warto byłoby stworzyć standardowy protokół gromadzenia danych monitoringu. Tak powstał protokół SNMP (ang. *Simple Network Management Protocol*).

Wbrew swojej nazwie SNMP jest w rzeczywistości dość skomplikowany. Określa hierarchiczną przestrzeń nazw dla zarządzanych danych oraz metody odczytu i zapisu tych danych na każdym urządzeniu sieciowym. SNMP definiuje również sposób wysyłania komunikatów powiadamiających o zdarzeniach („pułapek") przez zarządzane serwery i urządzenia („agenty") do stacji zarządzających.

Zanim zagłębimy się w tajniki SNMP, chcielibyśmy podkreślić, że terminologia z nim związana jest jednym z najbardziej beznadziejnych przypadków bełkotu technologicznego, z jakim można się zetknąć na rynku rozwiązań sieciowych. Niejednokrotnie standardowe nazwy pojęć i obiektów związanych z SNMP utrudniają zrozumienie tego, jaka jest ich rzeczywista rola.

Sam protokół jest jednak prosty; większość złożoności SNMP leży ponad warstwą protokołu w konwencjach tworzenia przestrzeni nazw oraz w zbyt wyszukanym słownictwie otaczającym SNMP jak skorupa ochronna. Pomijając jego mechanikę wewnętrzną, SNMP jest łatwy w obsłudze.

SNMP został zaprojektowany w taki sposób, aby mógł być wdrażany przez dedykowany sprzęt sieciowy, taki jak routery; w tym kontekście jest to słuszne rozwiązanie. Później SNMP został rozbudowany o możliwość monitorowania serwerów i systemów desktopowych, ale jego przydatność w tym zakresie zawsze budziła wątpliwości. Obecnie dostępne są znacznie lepsze rozwiązania (np. `collectd`; patrz podrozdział 28.6).

Sugerujemy, aby traktować SNMP jako niskopoziomowy protokół zbierania danych do użytku z urządzeniami o specjalnym przeznaczeniu, które nie obsługują niczego innego. Jak najszybciej wyprowadź dane ze świata SNMP i przekaż je platformie monitorującej ogólnego przeznaczenia w celu ich przechowywania i przetwarzania. SNMP może być ciekawą okolicą do zwiedzania, ale nie chciałbyś tam mieszkać!

Organizacja SNMP

Dane SNMP są uporządkowane w ustandaryzowanej hierarchii. Hierarchia nazw oparta jest na formacie MIB (ang. *Management Information Bases*), czyli ustrukturyzowanych plikach tekstowych opisujących dane dostępne za pośrednictwem SNMP. Pliki MIB zawierają opisy zmiennych określających pewne dane, do których można się odwoływać przy użyciu tzw. identyfikatorów obiektów (OID)[7]. Wszystkie obecnie stosowane urządzenia korzystające z SNMP obsługują strukturę w formacie MIB-II, zdefiniowanym w dokumencie RFC 1213. Producenci mogą jednak rozbudowywać MIB (i robią to), aby dodać więcej danych i wskaźników.

Identyfikatory obiektów istnieją w hierarchicznej przestrzeni nazw, gdzie węzły są numerowane, a nie nazwane. Jednak aby ułatwić odwoływanie się do nich, węzły mają również konwencjonalne nazwy tekstowe. Separatorem ścieżek jest tu kropka. Przykładowo OID odwołujący się do czasu nieprzerwanej pracy urządzenia ma wartość 1.3.6.1.2.1.1.3. Ten OID ma także czytelną dla człowieka (choć niekoniecznie „zrozumiałą bez dodatkowej dokumentacji") nazwę:

```
iso.org.dod.internet.mgmt.mib-2.system.sysUpTime
```

Tabela 28.3 przedstawia przykłady węzłów OID, które warto monitorować przy ocenie dostępności sieci.

Tabela 28.3. Wybrane OID z MIB-II

OID[a]	Typ	Zawartość
`system.sysDescr`	`string`	informacje systemowe: dostawca, model, system operacyjny itp.
`interfaces.ifNumber`	`int`	liczba obecnych interfejsów sieciowych
`interfaces.ifTable`	`table`	tabela charakterystyk każdego z interfejsów
`ip.ipForwarding`	`int`	1, jeśli system jest bramą, w przeciwnym wypadku 2
`ip.ipAddrTable`	`table`	tabela danych o adresaach IP (maski itp.)
`icmp.icmpInRedirects`	`int`	liczba otrzymanych przekierowań ICMP
`icmp.icmpInEchos`	`int`	liczba otrzymanych sygnałów ping
`tcp.tcpInErrs`	`int`	liczba otrzymanych błędów TCP

[a] Ścieżka względna w stosunku do *iso.org.dod.internet.mgmt.mib-2*.

Oprócz powszechnie obsługiwanego formatu MIB-II istnieją też formaty MIB dla różnego rodzaju interfejsów i protokołów sprzętowych, poszczególnych dostawców, różnych implementacji serwerów `snmpd`, a także dla poszczególnych produktów sprzętowych.

MIB to tylko schemat nazewniczy dla zarządzanych danych. Aby urządzenie MIB mogło być użyteczne, musi być obsługiwane przez kod działający po stronie agenta, który odwzorowuje rzeczywisty stan urządzenia na przestrzeń nazw SNMP.

Agenty SNMP działające w systemach Unix, Linux lub Windows posiadają wbudowaną obsługę MIB-II. Większość z nich można rozbudować o możliwość obsługi dodatkowych MIB oraz interfejsów za pomocą skryptów, które wykonają właściwą pracę polegającą na zbieraniu i przechowywaniu danych związanych z tymi definicjami MIB. Zobaczysz wiele tego rodzaju programów, będących pozostałością minionej epoki, kiedy to SNMP był jeszcze nowością.

[7] OID to po prostu wyszukane określenie na odwołanie do zarządzanej informacji.

Ale to już tylko dużo dymu bez ognia; obecnie nie powinieneś w ogóle uruchamiać agenta SNMP w systemie Unix, chyba że musisz uzyskać odpowiedź na najbardziej podstawowe pytania dotyczące konfiguracji sieci.

Operacje protokołu SNMP

Istnieją tylko cztery podstawowe operacje SNMP: `get`, `get-next`, `set` i `trap`.

`get` i `set` są głównymi operacjami odczytu i zapisu danych do węzła identyfikowanego przez określony OID. Operacja `get-next` służy do poruszania się w hierarchii MIB, ale może również odczytywać zawartość tabel.

Operacja `trap` (pułapka) to samodzielnie wysyłane, asynchroniczne powiadomienie od serwera (agenta) do klienta (zarządcy) o wystąpieniu interesującego zdarzenia lub stanu. Zdefiniowano kilka standardowych pułapek, w tym powiadomienia „właśnie się uruchomiłem”, raporty o awarii lub odzyskaniu połączenia sieciowego oraz komunikaty o różnych problemach z routingiem i uwierzytelnianiem. Mechanizm, za pomocą którego określa się miejsca docelowe komunikatów `trap`, zależy od implementacji agenta.

Ponieważ komunikaty SNMP mogą potencjalnie modyfikować informacje konfiguracyjne, potrzebny jest jakiś mechanizm bezpieczeństwa. Najprostsza wersja zabezpieczeń SNMP używa pojęcia łańcucha wspólnotowego (ang. *community string*), które jest tak naprawdę strasznie zagmatwanym określeniem hasła. Zazwyczaj jeden łańcuch wspólnotowy umożliwia dostęp w trybie tylko do odczytu, a drugi pozwala na zapisywanie[8]. W dzisiejszych czasach znacznie bardziej sensowne jest utworzenie mechanizmu zarządzania na bazie protokołu SNMPv3, który zapewnia większe bezpieczeństwo, w tym uprawnienia i kontrolę dostępu dla poszczególnych użytkowników.

Net-SNMP — narzędzia dla serwerów

W systemach Linux i FreeBSD najpopularniejszym pakietem implementującym SNMP jest Net-SNMP. Zawiera on agenta (`snmpd`), kilka narzędzi wiersza poleceń, serwer do odbierania komunikatów `trap`, a nawet bibliotekę do tworzenia aplikacji obsługujących SNMP.

Obecnie Net-SNMP stanowi przedmiot zainteresowania bardziej ze względu na swoje polecenia i biblioteki niż na to, że jest agentem. Został on przeniesiony do wielu różnych systemów uniksowych, dzięki czemu działa jako spójna platforma, na którą można pisać skrypty. Większość dystrybucji wydziela więc z agenta Net-SNMP własny pakiet, co sprawia, że łatwiej jest zainstalować same polecenia.

W systemach Debian i Ubuntu pakiety Net-SNMP noszą nazwy `snmp` i `snmpd`. Aby zainstalować tylko polecenia, wpisz `apt-get install snmp`.

W systemach Red Hat i CentOS analogiczne pakiety to `net-snmp` i `net-snmp-tools`. Zainstaluj polecenia przez `yum install net-snmp-tools`.

W Linuksie informacje konfiguracyjne trafiają do katalogu */etc/snmp*; zwróć uwagę na znajdujący się tam plik *snmpd.conf* i katalog *snmp.d*. Aby uruchomić demona, wpisz polecenie `systemctl start snmpd`.

[8] W wielu systemach łańcuch wspólnotowy ma domyślne ustawienie „public”. Nigdy tego tak nie zostawiaj; ustaw własne hasła dla obu łańcuchów, zarówno dla tego, który pozwala na odczytywanie, jak i tego, który umożliwia odczyt i zapis.

W systemie FreeBSD wszystko znajduje się w jednym pakiecie: `pkg install net-snmp`. Informacje konfiguracyjne są wprowadzane do katalogu */usr/local/etc/snmp*, który musisz samemu utworzyć. Możesz uruchomić agenta ręcznie poleceniem `service snmpd start` lub umieścić wpis:

```
snmpd_enable="YES"
```

w pliku */etc/rc.conf*, aby był uruchamiany wraz z systemem.

We wszystkich systemach, w których musisz uruchomić agenta SNMP, należy się upewnić, że port UDP 162 nie jest zablokowany przez zaporę sieciową.

Polecenia dostarczane przez Net-SNMP pomogą Ci się zapoznać z SNMP, a także świetnie się nadają do jednorazowych kontroli poszczególnych OID. Tabela 28.4 zawiera listę najczęściej używanych narzędzi.

Tabela 28.4. Narzędzia wiersza poleceń w pakiecie Net-SNMP

Polecenie	Działanie
`snmpdelta`	monitoruje zmiany wartości zmiennych SNMP w czasie
`snmpdf`	monitoruje przestrzeń dyskową na zdalnym hoście przez SNMP
`snmpget`	odczytuje wartość zmiennej SNMP podaną przez agenta
`snmpgetnext`	pobiera kolejną zmienną z sekwencji
`snmpset`	ustawia wartość zmiennej SNMP na agencie
`snmptable`	odczytuje tabelę zmiennych SNMP
`snmptranslate`	wyszukuje i wyświetla informacje o OID w hierarchii MIB
`snmptrap`	generuje komunikat `trap`
`snmpwalk`	przegląda kolejne wartości drzewa MIB, rozpoczynając od wskazanego OID

Podstawowe kontrole SNMP zazwyczaj wykorzystują połączenie `snmpget` i `snmpdelta`. Inne programy są pomocne, gdy chcesz zidentyfikować nowe OID do monitorowania za pomocą swojego wyszukanego narzędzia zarządzającego. Przykładowo `snmpwalk` rozpoczyna od określonego OID (domyślnie jest nim początek drzewa MIB) i wielokrotnie wywołuje na agencie operację `get-next`. Proces ten umożliwia uzyskanie kompletnej listy dostępnych OID oraz przypisanych im wartości.

W poniższym listingu przedstawiamy przycięty przykład działania polecenia `snmpwalk` na hoście *tuva* z systemem Linux. `secret813community` to łańcuch wspólnotowy, a opcja `-v1` wymusza proste uwierzytelnianie.

```
$ snmpwalk -c secret813community -v1 tuva
SNMPv2-MIB::sysDescr.0 = STRING: Linux tuva.atrust.com 2.6.9-11.ELsmp #1
SNMPv2-MIB::sysUpTime.0 = Timeticks: (1442) 0:00:14.42
SNMPv2-MIB::sysName.0 = STRING: tuva.atrust.com
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.3 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:11:43:d9:1e:f5
IF-MIB::ifPhysAddress.3 = STRING: 0:11:43:d9:1e:f6
```

```
IF-MIB::ifInOctets.1 = Counter32: 2605613514
IF-MIB::ifInOctets.2 = Counter32: 1543105654
IF-MIB::ifInOctets.3 = Counter32: 46312345
IF-MIB::ifInUcastPkts.1 = Counter32: 389536156
IF-MIB::ifInUcastPkts.2 = Counter32: 892959265
IF-MIB::ifInUcastPkts.3 = Counter32: 7712325
...
```

W tym przykładzie widzimy kilka ogólnych informacji o systemie, po których następują statystyki dotyczące interfejsów sieciowych: *lo0*, *eth0* i *eth1*. W zależności od tego, jakie drzewa MIB są obsługiwane przez agenta, którym zarządzasz, kompletny zrzut może się składać z setek wierszy. Prawdę mówiąc, pełna instalacja systemu Ubuntu skonfigurowanego do obsługi każdego MIB wyrzuca ponad 12 000 wierszy!

Jeśli przejrzysz drzewa MIB[9] w najnowszej wersji Net-SNMP w systemie Ubuntu, zobaczysz, że OID określający średnie obciążenie z ostatnich 5 minut to 1.3.6.1.4.1.2021.10.1.3.2. Jeśli chciałbyś zobaczyć to obciążenie dla lokalnego hosta (skonfigurowanego z łańcuchem wspólnotowym „public”), wpisz polecenie:

```
$ snmpget -v 2c -c public localhost .1.3.6.1.4.1.2021.10.1.3.2
iso.3.6.1.4.1.2021.10.1.3.2 = STRING: "0.08"
```

Wiele użytecznych modułów Perla, Ruby’ego i Pythona związanych z SNMP jest dostępnych w odpowiednich repozytoriach dla tych języków. Co prawda możesz pisać skrypty w formie poleceń Net-SNMP, ale zwykle łatwiej i prościej jest skorzystać z modułów dostosowanych do wybranego języka.

28.10. KRUCZKI I SZTUCZKI

Na przestrzeni lat zebraliśmy kilka wskazówek dotyczących maksymalizowania skuteczności monitoringu. Oto najważniejsze z nich:

- Unikaj zmęczenia spowodowanego monitoringiem. Dopilnuj, aby administratorzy systemów otrzymujący powiadomienia poza swoimi normalnymi godzinami pracy mieli regularne przerwy. Cel ten najlepiej osiągnąć w systemie rotacyjnym, gdzie w danym dniu albo tygodniu wzywane są dwu- lub kilkuosobowe zespoły, po czym zadanie przejmuje kolejny zespół. Nieprzestrzeganie tej zasady sprawi, że administratorzy staną się zgorzkniali i znienawidzą swoją pracę.
- Zdefiniuj okoliczności, które naprawdę wymagają uwagi 24 godziny na dobę przez 7 dni w tygodniu, i upewnij się, że informacje te zostały jasno przekazane Twojemu zespołowi monitorującemu, zespołom dyżurującym oraz obsługiwanym klientom lub jednostkom biznesowym. Sam fakt, że coś monitorujesz, nie oznacza, że administratorzy powinni być wzywani o 3:30 nad ranem, gdy jakaś wartość przekroczy normę. Wieloma kwestiami można się zająć w normalnych godzinach pracy.
- Eliminuj szumy monitorowania. Jeśli generowane są fałszywe alarmy lub powiadomienia o usługach niekrytycznych, poświęć czas na ich zatrzymanie i naprawienie. W przeciwnym razie, podobnie jak w bajce o chłopcu, który ostrzegał przed wilkiem, w końcu wszystkie zgłoszenia przestaną być traktowane z należytą powagą.

[9] Sprawdź na *mibdepot.com* lub zainstaluj pakiet `snmp-mibs-downloader`.

- Twórz procedury dla wszystkiego. Każde ponowne uruchomienie, każdy reset lub każdą procedurę naprawczą należy udokumentować w formie umożliwiającej podjęcie odpowiednich działań respondentowi, który nie zna bezpośrednio danego systemu. Skutki nieposiadania takiej dokumentacji są takie, że problemy nie będą szybko naprawiane, popełniane będą błędy, a w nagłych przypadkach trzeba będzie wzywać dodatkowy personel. Do przechowywania tego typu dokumentacji idealnie nadają się strony wiki.
- Monitoruj platformę monitorującą. Ta porada wyda Ci się oczywista, gdy przegapisz krytyczny przestój, ponieważ platforma monitorująca również przestanie działać. Ucz się na naszych błędach i upewnij się, że coś będzie obserwować Twoje czujne oczy.
- Przegapiłeś przestój z powodu czegoś, co nie było monitorowane? Upewnij się, że zostało to dodane, aby następnym razem wyłapać problem.
- I wreszcie ostatnia, choć być może najistotniejsza zasada: żaden serwer ani żadna usługa nie mogą wejść do środowiska produkcyjnego bez uprzedniego dodania ich do systemu monitorowania. I nie ma tu żadnych wyjątków.

28.11. ZALECANA LITERATURA

Hecht James, *Rethinking Monitoring for Container Operations*, *http://thenewstack.io/monitoring-reset-containers*. Wiele interesujących szczegółów dotyczących strategii i filozofii monitorowania kontenerów.

Turnbull James, *The Art of Monitoring*, Amazon Digital Services, Seattle 2016.

Dixon Jason, *Monitoring with Graphite: Tracking Dynamic Host and Application Metrics at Scale*, O'Reilly Media, Sebastopol 2017.

SKOROWIDZ

Pliki zostały uporządkowane alfabetycznie pod względem ostatniego elementu i w większości przypadków wymieniany jest tylko ten element. Przykładowo, aby znaleźć wpisy indeksu odnoszące się do pliku */etc/mail/aliases*, szukaj pod hasłem „aliases". Zostaliśmy zmuszeni do takiego postępowania przez producentów, którzy w każdym systemie ukrywają standardowe pliki w nowych i nietypowych katalogach.

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

Q

R

S

Ś

T

U

V

W

X

Y

Z

Ź, Ż

NOTATKI

Unix i Linux: niezawodność, wydajność i elastyczność na najwyższym poziomie!

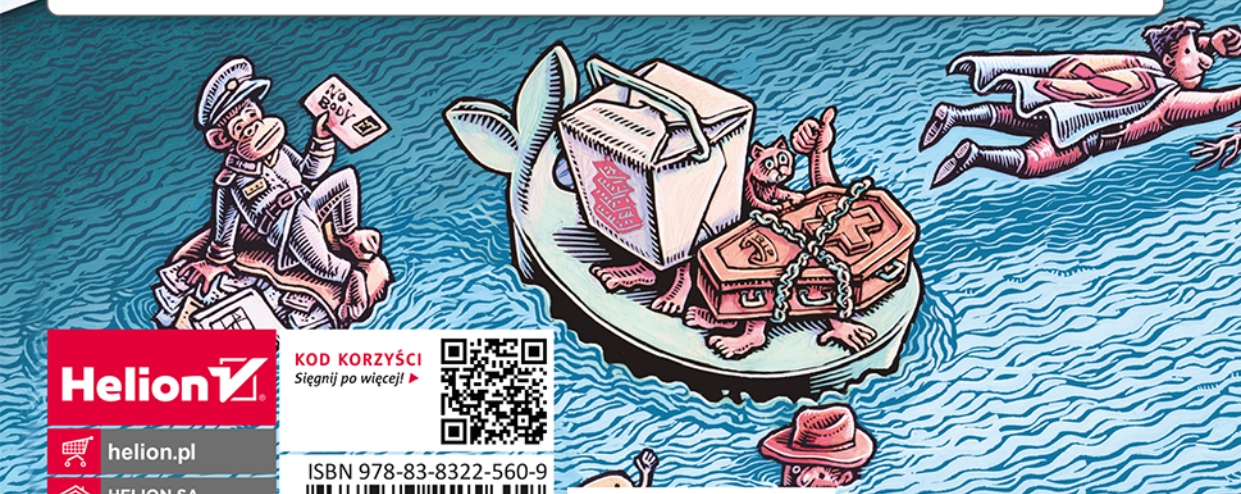
Mijają lata, a systemy Unix i Linux ciągle są gwarancją najwyższej niezawodności, wydajności i elastyczności. Ich potencjał jest niekwestionowany, a odporność na niestandardowe warunki zapewnia bezawaryjną pracę w różnych sytuacjach. To wszystko sprawia, że systemy te są wykorzystywane w dużych i złożonych organizacjach. Ich sieci wymagają profesjonalnego administrowania, co jest trudną i odpowiedzialną pracą. Łatwo skonfigurować pojedynczy system, znacznie trudniej jednak zapewnić stabilne działanie rozproszonej, opartej na chmurze platformy, narażonej na skokowe wzrosty popularności, rozbicia sieci i celowe ataki.

Ta książka jest kolejnym, uzupełnionym i zaktualizowanym wydaniem kultowego podręcznika dla profesjonalnych administratorów zarządzających środowiskami produkcyjnymi w korporacjach czy urzędach. Znajdziesz tu obszerne wyjaśnienie takich zagadnień jak instalacja systemu, skrypty powłoki, kontrolowanie procesów czy konfiguracja uprawnień. Dowiesz się, jak zarządzać użytkownikami, przestrzenią dyskową, zadaniami okresowymi oraz backupami. Przestudiujesz zagadnienia sieciowe, a zwłaszcza kwestie bezpieczeństwa i reakcji na incydenty. W tym wydaniu znajdziesz m.in. omówienie demonów zarządzania systemem, zasady zarządzania kontem użytkownika root, techniki kontroli wersji za pomocą narzędzia Git, kwestie związane z zarządzaniem sieciami w chmurze, tworzeniem i utrzymywaniem centrów danych, opis metodologii DevOps i wiele innych!

Najciekawsze zagadnienia:

- podstawy administrowania i obowiązki administratora
- system plików i zarządzanie oprogramowaniem
- rejestrowanie zdarzeń
- zarządzanie jądrem systemu i sterownikami
- sieci i sprzęt sieciowy
- zapewnianie wydajności i niezawodności sieci
- metodologie i reguły w IT

Evi Nemeth była matematykiem, kryptografem i wiodącą współautorką książek uznawanych za biblie administratorów systemów. Była również autorytetem w dziedzinie sieci. W 2013 roku zaginęła bez wieści, gdy żeglowała po Morzu Tasmana.
Garth Snyder jest inżynierem. Pracował w firmach NeXT i Sun.
Trent R. Hein jest pasjonatem bezpieczeństwa informatycznego i automatyzacji. Lubi piesze wędrówki, narty, muzykę bluegrass, psy i gramatykę.
Ben Whaley jest założycielem WhaleTech, niezależnej firmy doradczej. Został uhonorowany przez firmę Amazon jako jeden z pierwszych bohaterów społeczności AWS.
Dan Mackin jest zdeklarowanym użytkownikiem Linuksa i innych technologii open source. Uwielbia jeździć na nartach, żeglować i spędzać czas z żoną i psem.



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 978-83-8322-560-9
9 788383 225609
Cena: 199 zł
Pearson
Addison-Wesley