PŁYTA DVD

Tails 3.4
Linux zapewniający pełną anonimowość w sieci

Menedżer haseł na USB
i wiele innych przydatnych narzędzi

Bezpieczne uruchamianie
Zabezpieczamy system za pomocą Secure Boot

PŁYTA W WYDANIU DRUKOWANYM

LINUX MAGAZINE

LUTY 2018 – NUMER 2 (168)
CENA: 17,90 ZŁ (23% VAT)

Bezpieczne uruchamianie
Kontrolujemy proces uruchamiania systemu

Sztuczki z UEFI
Dodajemy własne aplikacje do oprogramowania sprzętowego

5 narzędzi do automatycznego wykonywania kopii zapasowych

Radio FM
na Raspbery Pi

Ataki DDoS
Bronimy się przed zmasowanymi atakami sieciowymi

Stacer
Jak utrzymać porządek w systemie?

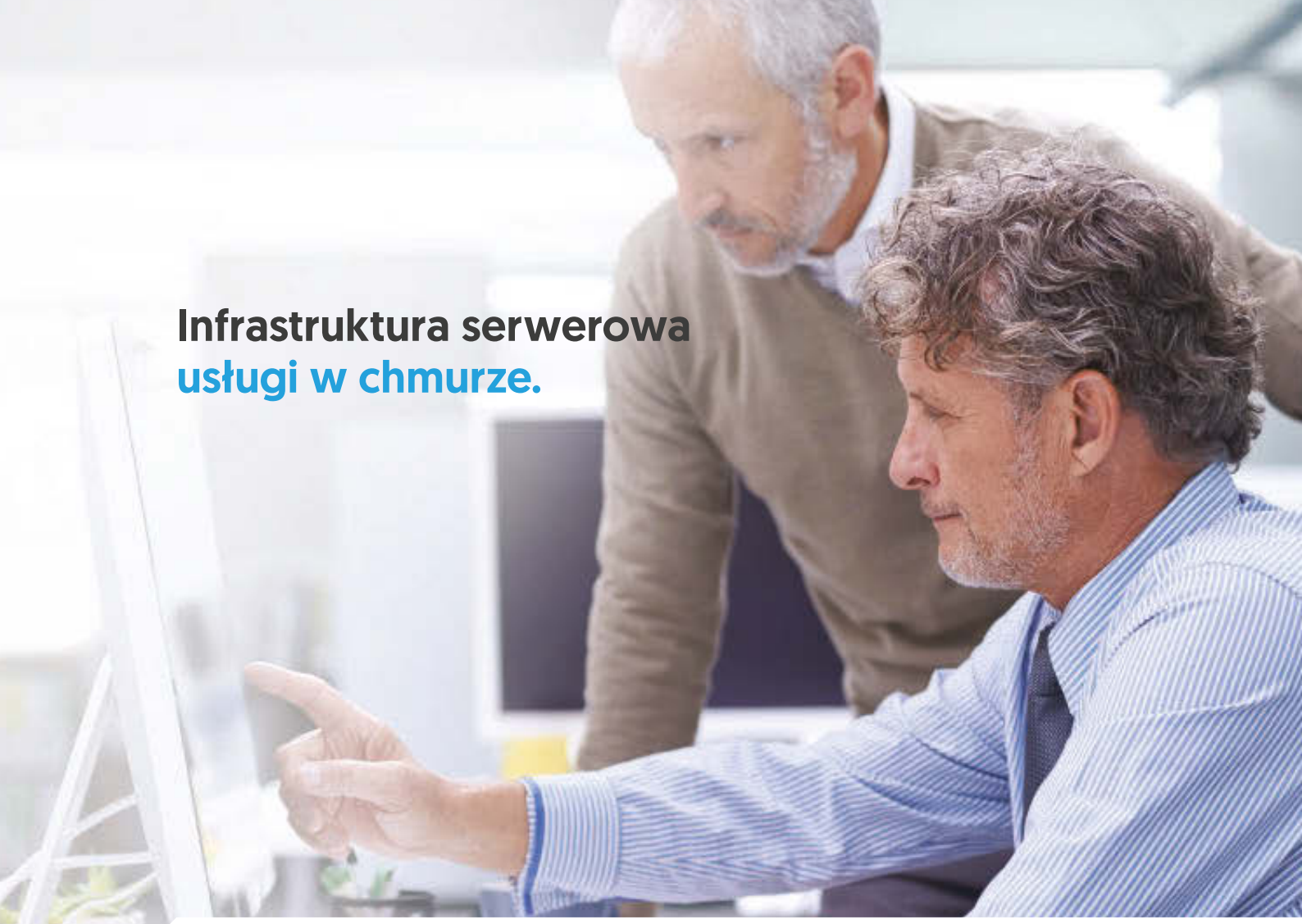
LINUXVOICE

- Marketing Wolnego Oprogramowania
- Digitalizujemy kolekcję muzyki
- Ulepszamy domowy zestaw stereo

Perełki FOSS
- qutebrowser 1.0
- Storyboarder
- CoreFreq
- Fragment synthesizer

Samouczek
- FFmpeg

E-WYDANIE



Infrastruktura serwerowa usługi w chmurze.

Usługi w chmurze dla biznesu i developerów

UniCloud - autoskalowalna w górę i w dół, nowoczesna chmura do zadań specjalnych, oparta o technologię SSD. Szybka, niezawodna, automatyczna. Umożliwia łatwą implementację i zarządzanie wieloma środowiskami.

Apache / Nginx / Tomcat - przygotuj serwer aplikacji w mniej niż 90 sekund!

Tylko do końca marca, dla każdego czytelnika Linux Magazine **100 zł** do wykorzystania na platformie UniCloud.

1. Wejdź na stronę www.unicloud.pl/kupon
2. Wpisz kod: LINUX
3. Wykorzystaj kupon o wartości 100 zł

Centrum Danych Asseco Data Systems S.A.

Infrastruktura serwerowa

Bezpieczeństwo danych

Wsparcie projektowe

centrumdanych.assecods.pl

Centrum Danych by asseco

Osobista chmura

Dwa miesiące temu wspomniałem w tym miejscu o Subutai: otwartym oprogramowaniu P2P stworzonym przez firmę OptDyn. Od tamtej pory otrzymuję pytania o sposoby wykorzystania Subutai w pracy i w domu.

Obecnie wiele osób przechowuje swoje dane w komercyjnych chmurach, takich jak Amazon, Google czy Microsoft Azure. Choć w ten sposób mogą mieć do nich dostęp z wielu różnych maszyn w danej sieci i poza nią (bez względu na to, czy to sieć domowa, czy firmowa), oznacza to, że koszty komunikacji mogą rosnąć przy dostępie do danych spoza infrastruktury chmurowej.

Wyobraźmy sobie, że pracujemy w domu, przechowując muzykę, zdjęcia i dane na jednej z komercyjnych chmur. Płacimy za przesyłanie danych do chmury, a potem drugi raz za ich pobranie. Przy większej ilości danych koszty takiego rozwiązania mogą być znaczne, więc wiele osób konfiguruje w domu czy w pracy system NAS, zyskując bezpośredni dostęp do danych z sieci lokalnej. Niestety, oznacza to, że dane nie są w równie prosty sposób dostępne spoza tej samej sieci, w której działa NAS. Jeśli jednak skonfigurujemy Subutai i połączymy go z NAS-em, możemy uzyskać dostęp do danych znajdujących się w NAS-ie bez konieczności płacenia dostawcy usług chmurowych za przesyłanie danych; zazwyczaj też czas dostępu jest znacznie krótszy. Kiedy zaś jesteśmy poza siecią lokalną, Subutai nadal w przezroczysty sposób zapewnia nam dostęp do danych.

Niektórzy korzystają też z chmur do wykonywania kopii zapasowych swoich danych. Dla niewielkich ilości danych taka usługa może być darmowa, jednak dane szybko rozrastają się i ostatecznie musimy co miesiąc płacić określoną kwotę – przy terabajtach danych może być ona stosunkowo wysoka. Jeżeli jednak mamy zaufanego znajomego czy członka rodziny z systemem NAS, możemy skonfigurować mechanizm wzajemnego wykonywania kopii zapasowych przy użyciu Subutai. W razie katastrofy (pożar, powódź, kradzież) możemy pójść do niego z nowym dyskiem, skopiować dane i przywrócić system NAS. Jeśli nie zależy nam na czasie, możemy to nawet zrobić przez Internet bez wychodzenia z domu.

Inny scenariusz to przygotowanie osobistego pendrive'a uruchamiającego ulubioną dystrybucję GNU/Linuksa z opcją trwałego przechowywania danych na Subutai. Uruchomiwszy dystrybucję, nawiązujemy połączenie z Subutai i uzyskujemy dostęp do środowiska chmurowego. Po zamknięciu systemu nasze dane przechowywane są w naszej prywatnej chmurze Subutai, natomiast dzięki opcji trwałego przechowywania danych nasze ustawienia zostaną zachowane – gotowe na kolejną sesję z Subutai. Dla osób, które – podobnie jak ja – często podróżują, oznacza to, że wystarczy zabrać ze sobą jednego pendrive'a, by móc pracować na dowolnym komputerze na świecie. Nasze dane spoczywają bezpiecznie w chmurze prywatnej, z dala od wścibskich oczu celników w niektórych krajach.

A co z firmami telekomunikacyjnymi, które czasem czują się „ominięte" przez korporacje oferujące usługi chmurowe? Subutai daje takim firmom możliwość stworzenia nowego rozwiązania na bazie oprogramowania open source, umożliwiającego klientom korzystanie z wolnych zasobów w centrach danych. To samo dotyczy zresztą firm hostingowych i ISP.

Pozostałym firmom Subutai daje możliwość konsolidacji poszczególnych zasobów wewnętrznych, co pozwala zredukować koszty związane z korzystaniem zewnętrznych zasobów telekomunikacyjnych i chmurowych. Nie trzeba zupełnie rezygnować z usług firm zewnętrznych, ale można znacznie zredukować zapotrzebowanie na ich usługi. W przypadku tzw. wrażliwych danych można użyć prywatnej chmury Subutai. Mówiąc „firmy", mam na myśli również organizacje i instytucje państwowe, takie jak szpitale.

Kiedy pierwszy raz w życiu poleciałem do Brazylii, odwiedziłem Uniwersytet w Sao Paulo (USP), gdzie po raz pierwszy zobaczyłem na żywo superkomputer w postaci klastra Beowulfa. Jednym z jego zastosowań była diagnostyka obrazów mammograficznych, czyli zdjęć rentgenowskich wykorzystywanych do wykrywania raka piersi u kobiet. Program działający na stacji roboczej SPARC przetwarzał jedno zdjęcie w dwadzieścia godzin, podczas gdy Beowulfowi to samo zadanie zajmowało jedynie dziesięć minut, co oznaczało, że wyniki badań były dostępne praktycznie od ręki i w razie konieczności można było od razu zlecić dalsze testy.

Naukowcy nie zamierzali namawiać każdego szpitala na zakup Beowulfa, lecz tak skonstruować klaster, by rozłożyć obciążenie na wszystkie laptopy, stacje robocze i małe serwery w szpitalu, by uzyskać ten sam efekt. Dzięki Subutai można z łatwością zaimplementować tego typu konfigurację – jeśli uda się w ten sposób uratować życie choć jednej kobiety, będzie to opłacalne przedsięwzięcie. ■■■

Linux Magazine jest miesięcznikiem specjalistycznym wydawanym na licencji Linux New Media USA, LLC, we współpracy z Computec Media GmbH, Fürth, Niemcy.

Wydawca Wiedza i Praktyka Sp. z o.o.

Redaktor Naczelny

Artur Skura, *askura@linux-magazine.pl*

Kierownik grupy tematycznej:

Alina Sulgostowska, *asulgostowska@wip.pl*

Korespondenci i współpracownicy

Erik Bärwaldt, Chris Binnie, Zack Brown, Bruce Byfield, Karsten Günther, Marcel Hilzinger, Klaus Knopper, Christoph Langner, Jeff Layton, Martin Loschwitz, Patrick Neef, Dimitri Popov, Thorsten Scherf, Ferdinand Thommes

Opracowanie graficzne, skład i przygotowanie do druku

Raster studio, Norbert Bogajczyk, *studio@rasterstudio.pl*

Projekt okładki

Lori White

Reklama

Agnieszka Zduńczyk, *jzdunczyk@wip.pl*
kom.: +48 507 124 623

Prenumerata

Andrzej Chłodziński, *redakcja@linux-magazine.pl*
prenumerata@linux-magazine.pl

Ceny prenumeraty:

kwartalna (3 numery) 75 zł
półroczna (6 numerów) 125 zł
roczna (12 numerów) 225 zł
dwuletnia (24 numery) 355 zł

Szczegóły: *http://linuxmagazine.pl/index.php/subscribe*

Ceny e-prenumeraty:

kwartalna (3 numery) **49 zł**
półroczna (6 numerów) **99 zł**
roczna (12 numerów) **149 zł**
dwuletnia (24 numery) **270 zł**

Licencje rozszerzone
2 do 5 użytkowników
półroczna (6 numerów) **115 zł**
roczna (12 numerów) **200 zł**
dwuletnia (24 numery) **325 zł**
nieograniczona roczna (12 numerów) **2000 zł**

Ceny prenumeraty łączonej (wersja papierowa i cyfrowa):

kwartalna (3 numery) **90 zł**
półroczna (6 numerów) **150 zł**
roczna (12 numerów) **265 zł**
dwuletnia (24 numery) **430 zł**

Szczegóły: *http://linuxmagazine.pl/index.php/eprenumerata*

Zamówienia i obsługa prenumeraty:

tel.: +48 22 429 43 05
faks: +48 22 617 60 10
prenumerata@linux-magazine.pl

Linux Magazine

ul. Łotewska 9a, 03-918 Warszawa
www.linux-magazine.pl,
tel.: +48 22 429 43 05
faks: +48 22 617 60 10

ISSN 1732-1263; Nakład 5000 egz.

W WYDANIU

W tym miesiącu przyjrzymy się procesowi uruchamiania Linuksa i omówimy kilka narzędzi i mechanizmów zwiększających bezpieczeństwo tego procesu, w tym program rozruchowy Shim i metody wykorzystania modułów TPM znajdujących się na płytach głównych większości produkowanych współcześnie komputerów.

Warto też zwrócić uwagę na:

Pokażemy też, jak edytować pliki audio i wideo za pomocą narzędzi Audacity i FFmpeg, zaś wśród **Perełek FOSS** wyróżniliśmy minimalistyczne przeglądarki i edytory tekstu.

NOWOŚCI

6 LINUX

9 JĄDRO

RAPORT

28 Automatyczne kopie zapasowe

Wykonywanie kopii zapasowych jest zadaniem, który wielu użytkowników uważa za nudny obowiązek. Dlaczego więc go nie zautomatyzować? Przedstawiamy kilka interesujących narzędzi, które nam w tym pomogą.

TEMAT NUMERU

LINUXVOICE

KNOW-HOW

MAKERSPACE

Tails 3.4

Bezpieczny Linux do ochrony prywatności w Internecie

- **Pełna anonimowość w sieci**
- **Bezpieczne narzędzia komunikacyjne**
- **Brak modyfikacji na dysku twardym**
- **Wersja zabezpieczona przed atakami Spectre i Meltdown**

STRONA 12

NOWOŚCI

01 KUBECON W TEKSASIE

Kubernetes, masowo wybierany przez klientów w ciągu ostatnich trzech lat, został w końcu dyżurnym Linuksem dla usług chmurowych. Pierwsze wydanie Kubernetesa ogłoszono w 2014 roku. Teraz korzystają z niego: Google (jego twórcy), Microsoft i AWS, a więc wszyscy trzej najwięksi dostawcy usług w chmurze. Zaczyna z niego korzystać nawet Docker, obok swojego własnego orkiestratora o nazwie Swarm. Cloud Foundry zastosowała Kubernetesa w roli środowiska dla klastrów, Cloud Foundry Container Runtime, natomiast twórcy OpenStack już używają go do wdrożeń OpenStacka jako aplikacji. Swoje kubernetesowe dystrybucje oferują też wszyscy wydawcy głównych dystrybucji: Red Hat, SUSE i Canonical.

Właśnie wzrost popularności i liczby instalacji Kubernetesa był tematem KubeCon, który odbył się między 6 a 8 grudnia 2017 r. w Austin w Teksasie. W trakcie konferencji Oracle uwolnił źródła swoich związanych z Kubernetesem narzędzi do bezserwerowego wdrażania aplikacji i zarządzania wieloma chmurami.

Microsoft ogłosił, że nowe wydanie Azure wniesie do społeczności Kubernetesa więcej funkcjonalności bezserwerowych i DevOps, natomiast Bitnami wdrożyło nową klastrową konsolę do wdrażania aplikacji Kubernetesa (Kubeapps).

Społeczność Kubernetesa ogłosiła wydanie 1.0 CordeDNS, narzędzia obsługującego DNS w Kubernetesie. Do macierzystej fundacji Kubernetesa, Cloud Native Computing Foundation, ze statusem złotych członków dołączyły firmy JFrog i Baidu.

02 DELL WYŁĄCZY DZIURAWE IME OD INTELA

Intelowski podsystem na chipsetach procesorów, IME (Intel vPro Management Engine), spotkał się z falą krytyki po tym, jak badacze bezpieczeństwa znaleźli w nim błędy pozwalające atakującemu zdalnie przejąć kontrolę nad podatnymi komputerami.

„Exploit umożliwia atakującemu uzyskanie pełnej kontroli nad firmowymi komputerami, nawet jeśli są wyłączone (ale podpięte do gniazdka sieciowego). Pragniemy, aby ujawnienie tej podatności zwiększyło znajomość problemów z bezpieczeństwem firmware i pomogło unikać takich incydentów w przyszłości" – komunikuje Embedi, firma zajmująca się bezpieczeństwem i odkrywca błędu.

Intel nie udostępnia żadnych informacji o swoich tajemniczych zamkniętych technologiach – moduły Management Engine są bliżej sprzętu niż system operacyjny, a użytkownicy nie mają do nich dostępu ani możliwości sterowania nimi.

Skoro Intel nie dzieli się informacjami o swojej technologii, to producenci komputerów i użytkownicy nie mają możliwości wykrycia ani załatania takich podatności. Póki co Dell jako jedyny z producentów próbuje chronić użytkowników, wyłączając IME we wszystkich nowych komputerach. Aby włączyć taką usługę, użytkownicy będą musieli dodatkowo zapłacić.

Dell wypowiedział się dla portalu ExtremeTech: „Przez kilka lat będziemy oferowali opcję konfiguracji polegającą na wyłączeniu Intel vPro Management Engine (ME) na wybranych komercyjnych platformach klienckich. Konfiguracja, zwana «Intel vPro – ME inoperable», jest dostępna na zamówienie na Dell.com. Niektórzy klienci biznesowi zwracali się do nas z prośbą o taką możliwość, więc odpowiadając na konkretne potrzeby, oferujemy usługę fabrycznego wyłączenia Management Engine. Opcja ta nie była wcześniej dostępna dla masowych odbiorców, ponieważ może powodować wyłączenie również innych funkcji".

W ślad Della pójdą inni producenci komputerów, szczególnie takich z preinstalowanym Linuksem. Dell jest największy w branży, a jeśli podatny silnik zaczną wyłączać również inni, Intel może być zmuszony do otworzenia źródeł technologii lub przynajmniej pokusić się o nieco przejrzystości.


© Chode, 123RF.com

03 DOBRE RADY LINUSA TORVALDSA DLA EKSPERTÓW OD BEZPIECZEŃSTWA

Linus Torvalds, twórca jądra Linuksa, nie przepada za społecznością zajmującą się bezpieczeństwem – jego zdaniem te problem to zwykłe błędy, które ktoś wykorzystał. „Nie ufam ludziom od bezpieczeństwa, mogą zrobić coś szalonego" – powiedział Torvalds w odpowiedzi na prośbę o włączenie łaty od jednego z najważniejszych developerów jądra, Keesa Cooka.

Tym razem Torvaldsa rozzłościło to, że łata Keesa mogłaby coś zepsuć. Kees dodał do niej tryb fallback i napisał: „Łata od jakiegoś czasu leży na -next bez większych problemów, ale niedawno dowiedzieliśmy się o brakujących białych listach, więc doszedł tryb fallback, żebyśmy na pewno niczego nie popsuli. Za jedno lub dwa wydania planuję usunąć fallback".

Torvalds odmówił włączenia łaty do kodu jądra i skomentował: „Bardziej strawne mogłoby być żądanie pull od Ciebie wprowadzające tę samą infrastrukturę, ale takie, które na pewno niczego nie zepsuje".

Kees odpisał: „Właśnie dlatego dodałem tryb fallback – kvm i sctp (ipv6) zostały zauważone dopiero pod koniec cyklu rozwojowego, więc zasmuciło mnie, że odbyły się już wszystkie potrzebne testy. Chciałem zagwarantować, że seria trafi na swoje miejsce i niczego nie zepsuje z powodu ewentualnych przeoczonych białych list".

Torvalds zareagował: „Nie jestem ani trochę zainteresowany zabijaniem procesów. Jedyny proces, jaki mnie obchodzi, to development, kiedy to znajdujemy błędy i je naprawiamy".

Oprócz zwykłych przepychanek tym razem mamy też cenną radę dla specjalistów bezpieczeństwa. Zdaniem Torvaldsa priorytetem powinno być debugowanie jądra i dokładanie starań, by wydanie za rok było lepsze niż wydanie dzisiejsze. Torvalds odrzuca popularną koncepcję zabijania procesów z powodu błędów. „(...) prace nad uszczelnieniem nie powinny się kończyć, a zaczynać procedurą «ostrzeżmy wszystkich, że coś wygląda groźnie, a za rok, po wystarczająco długim ostrzeganiu, będziemy pewni, że wyłapaliśmy wszystkie normalne przypadki». Dopiero wtedy można podejmować drastyczniejsze kroki. Skończmy z idiotyczną postawą «zabij, póki to widać, pytania będą później»".

04 LICENCJA GPLV3 POMOŻE TYM, KTÓRZY JĄ ŁAMIĄ

Red Hat współpracuje z największymi firmami technologicznymi, w tym Facebookiem, Google i IBM-em, aby ułatwić podmiotom łamiącym licencję GPL spełnienie jej warunków. Wymienione spółki, pomagając innym, stosują postanowienia naprawcze GNU GPLv3.

Zgodność z licencją jest jedną z największych bolączek korzystania z komponentów open source w produktach komercyjnych. Z jednej strony organizacje takie jak Linux Foundation podejmują wiele różnych starań, by firmy korzystały z oprogramowania open source, nie martwiąc się o zgodność z warunkami licencji. Z drugiej strony są przypadki takie jak agresywny atak Software Freedom Conservancy na firmę VMware, co zaszkodziło wzajemnej współpracy i spornym projektom open source. Co gorsza, takie działania prawne wysyłają w świat komunikat, że porywanie się na open source kończy się kontuzją.

Firmy nie łamią przecież warunków licencji celowo. „Większość przypadków wynika z błędu, a nie złej woli. Egzekwowanie Copyleft powinno pomóc takim dystrybutorom przekształcić się w pomocnych uczestników projektów wolnego oprogramowania, z którego sami korzystają" – powiedział Joshua Gay z Free Software Foundation.

Takim firmom trzeba jednak dać możliwość poprawienia zgodności. Licencja GPLv2, jedna z najbardziej znanych licencji copyleft, przewiduje samoczynne trwałe zerwanie licencji w momencie złamania jej warunków. To dość brutalne podejście, które zniechęcało podmioty do współpracy i prowadziło do kolejnych konfliktowych zabiegów prawnych, na których nie korzystał nikt poza prawnikami. Linus Torvalds powiedział kiedyś, że kiedy tylko wkraczają prawnicy, „przegraliśmy".

Wprowadzając licencję GPLv3, Free Software Foundation dała użytkownikom możliwość popracowania nad stwierdzonymi przekroczeniami. →

04

Jest to poprawa w porównaniu do surowych warunków łamania licencji zawartych w GPLv2. Licencja GPLv3 daje podmiotom łamiącym licencję po raz pierwszy możliwość przywrócenia wynikających z niej praw w chwili, gdy wykroczenia zostaną zażegnane. Ma to na celu tworzenie liczniejszych okazji do współpracy i wybieranie rozstrzygnięć polubownych zamiast agresywnych pozwów.

Red Hat, Facebook, Google i IBM będą angażować się w rozciągnięcie podejścia wprowadzanego licencją GPLv3 na błędy zgodności w kodzie objętym do tej pory licencjami GPLv2 oraz LGPLv2.1 i v2.

Wraz z przyjęciem tego zrównoważonego rozwiązania firmom będzie wygodniej stosować komponenty open source w swoich produktach i nie będą musiały obawiać się pozwów za nieumyślne złamanie warunków.

Red Hat we wpisie na swoim blogu wyraził nadzieję, że „duże ekosystemy projektów na licencjach GPLv2 i LGPLv2.x znacznie skorzystają na tych przyjaźniejszych warunkach zrywania licencji, pochodzących z GPLv3". Takie działanie Red Hata należy zdecydowanie pochwalić.

05

JĄDRO LINUKSA – JEST WERSJA 4.14

Twórca Linuksa, Linus Torvalds, ogłosił 12 listopada 2017 r. wydanie jądra w wersji 4.14. Przewidywany termin był wcześniejszy, ale wydanie opóźniło się z powodu łaty na AppArmor powodującej regresję. Torvalds zaatakował developera z Canonical, który znalazł regresję związaną z AppArmor, ale skomentował, że nie ma czym się przejmować.

Torvalds odpowiedział: „W przypadku jądra pod pojęciem regresji rozumiemy brak powtórzenia takich samych efektów działania jądra przy takiej samej przestrzeni użytkownika. Regresję powoduje jądro, więc twoje próby zepchnięcia jej gdzie indziej to podrabiany szajs. Właśnie takie szmiry zrażają mnie do Twoich opinii i kodu. Jeśli nie chcesz przyznać, że regresję wywołał Twój commit 651e28c5537a («apparmor: add base infrastructure for socket mediation»), to już wolę zrezygnować z Twoich commitów".

Torvalds zdecydował się na opóźnienie wydania jądra, aby nie dopuścić do ostatecznego zaimplementowania regresji.

Jądro Linuksa w wersji 4.14 ma zostać następną wersją z długoterminowym wsparciem (LTS). Jak mówi Greg Kroah-Hartman, opiekun stabilnej gałęzi jądra: „Wersja 4.14 jest już wydana oficjalnie i prawdopodobnie będzie kolejną wersją LTS wspieraną przeze mnie stabilnymi łatami przenoszonymi wstecznie przez co najmniej dwa lata. No chyba że to wydanie okaże się kotem w worku z okropnymi problemami. Jeśli tak, to zastrzegam sobie prawo wyboru innego jądra, ale cykl wydań ostatnio idzie nam gładko, więc może nie będzie to konieczne (oj, chyba wykrakałem...)".

Z ciekawszych nowości w tym wydaniu warto wymienić zintegrowaną obsługę HDMI-CEC dla Raspberry Pi, dzięki której użytkownicy kilku urządzeń opartych na „malinkach" będą mogli sterować nimi jednym pilotem, a także duże ulepszenia wydajności KVM, Xen i Hyper-V. W tym wydaniu ulepszono także obsługę EFI pod kątem bezpieczeństwa i stabilności.

R E K L A M A

NOWOŚCI
JĄDRO

Nowości w jądrze

Kronikarz Zack Brown informuje o poglądach, wydarzeniach, dylematach i innych nowościach w społeczności programistów jądra. Zack Brown

Obsługa I3C

Boris Brezilon opublikował łatki implementujące część głównej infrastruktury I3C. Jest to kompleksowe uaktualnienie protokołu I2C, służącego komunikacji przez porty szeregowe, z którego korzysta duża grupa czujników, ponieważ jest to prosty, dwużyłowy interfejs. Jednak, podczas gdy prostota sprawia, że ilość urządzeń sensorycznych rośnie, zarządzanie rosnącym wolumenem danych przesyłanych tą drogą i ilością przerwań potrzebnych do kolejnych urządzeń zaczyna mieć duże znaczenie. Dlatego właśnie powstał I3C.

Rozwiązanie Borisa jest wstecznie kompatybilne z I2C, dzięki czemu nie sprawia problemów użytkownikom. Jednak w niektórych kwestiach autor musiał pójść na kompromis i dlatego korzystanie z jego API może sprawiać problemy. Dodatkowo pozostawił niezaimplementowaną część API I3C, chociaż planuje uzupełnić tę lukę w przyszłości.

ZACK BROWN

Lista dyskusyjna poświęcona rozwojowi jądra jest głównym narzędziem komunikacyjnym programistów jądra. Ruch na niej jest ogromny – dochodzi do dziesięciu tysięcy listów tygodniowo. Pozostawanie z wszystkim na bieżąco to zadanie bardzo trudne dla zwykłych śmiertelników. Zack Brown jest jedną z niewielu osób, która uważnie śledzi wszystkie dyskusje.

Jednym ze wspomnianych kompromisów jest wymaganie, aby operacje związane z oprogramowaniem użytkownika były nieatomowe (czyli dany proces może zostać przerwany przez coś innego). Jest to pewnie niedogodność, gdyż kod użytkownika musi znać swój aktualny stan w czasie wywołania API I3C. Boris zaznaczył jednak, że może to zmienić. Po prostu takie rozwiązanie było szybsze.

Jednym z brakujących fragmentów API jest obsługa urządzeń podłączanych podczas pracy komputera (hot plugging), co dla wielu może skreślać nowe rozwiązanie. Oczywiście, z czasem kod I3C będzie uzupełniany i brakujące API zostaną dodane.

Wolfram Sang przyjrzał się kodowi źródłowemu, ale nie miał obiekcji i zatwierdził łatki.

Arnd Bergmann zapytał, dlaczego Boris stworzył cały nowy podsystem dla I3C, zamiast po prostu rozwinąć istniejący I2C, tak aby wspierał też I3C. Boris odpowiedział:

„I3C i I2C są zupełnie inne. Nie mówię o warstwie fizycznej, ale o tym jak magistrala ma być obsługiwana przez warstwę programową. Uważam, że I3C bliżej jest do magistrali wykrywających urządzenia automatycznie, takich jak USB czy SPI.

Wszystkie urządzenia I3C mogą być wykryte i nie muszą być identyfikowane na poziomie płyty (z wykorzystaniem drzewa urządzeń (DT), ACPI czy czegokolwiek innego). Ponadto, niektóre z nich można podłączyć do działającego sprzętu i, co najważniejsze, wszystkie identyfikują się w czasie procedury wykrycia (nazywanej DAA w I3C).

Istnieje coś analogicznego do «klasy urządzenia». W świecie I3C nazywa się to DCR (Device Characteristic Register, czyli Rejestr Cech Urządzenia), ale odgrywa tę samą rolę: jest to zbiór standardowych interfejsów, z którymi urządzenia muszą być zgodne, jeśli chcą być kompatybilne z DCR ID (jak przyspieszeniomierz, żyroskop itp.) [...]

Urządzenia udostępniają również 48-bitowe, składające się z mniejszych pól, tymczasowe ID. Dwa z tych pól są szczególnie interesujące: ID producenta i ID części, które można porównać do dostawcy i ID produktu w świecie USB.

Podane informacje (DCR, ID producenta i ID części) mogą zostać wykorzystane przy dopasowaniu sterowników, zamiast nazw, jak to odbywa się w urządzeniach I2C.

Jak więc widać, współpraca z szyną I3C jest zupełnie inna niż z I2C"

Boris dodał: „Oczywiście mogę przenieść cały kod do *drivers/i2c/*, ale nie zmieni to faktu, że magistrale I3C i I2C są całkowicie różne i niewiele je łączy".

Jednak Arnd chciał kontynuować dyskusję. Stwierdził, że istnieją argumenty, aby nie rozszerzać I2C, tak aby obejmowało też I3C, uważa jednak, że są też przesłanki, by jednak to zrobić, nawet jeśli powstanie lekki bałagan. Powiedział: „Odzwierciedlenie fizycznej hierarchii w oprogramowaniu ma wiele zalet i jeśli urządzenia I2C i I3C mogą być podłączone do tej samej magistrali, dobry model powinien przedstawiać je na tym samym poziomie. Odnosi się to zarówno do jądra (w *sysfs* i strukturach danych), jak i do drzewa urządzeń (zakładając, że urządzenia I3C należy w ogóle w ten sposób opisywać). Oba rozwiązania nie muszą korzystać z tego samego modelu, ale łatwiej by było, gdyby tak robiły".

Następnie dodał: „W przeszłości dyskutowaliśmy już, czy I2C i SPI powinny był połączone pod jednym typem *bus_type*, ponieważ wiele urządzeń może być podłączanych do obu interfejsów. Jeśli okazuje się, że urządzenia I3C dość

NOWOŚCI

często wspierają tryb awaryjny I2C, posiadanie tej samej wartości *bus_type* może w znacznym stopniu uprościć sterowniki, gdyż wymagana będzie jedynie pojedyncza struktura *i2c_driver*. Większe skomplikowanie po stronie podsystemu, zrównoważone więc będzie przez uproszczenie sterowników".

Boris stwierdził, że nie dostrzega korzyści płynących z takich rozwiązań i wyraził zdziwienie na temat idei połączenia I2C i SPI. Poprosił Arnda o wyjaśnienie, na co ten odpowiedział: „nigdy nie dokonaliśmy tego połączenia, więc pewnie okazało się, że koszt tej operacji był większy niż korzyści". Jednak określił, że ogólnym celem jest uproszczenie opcji konfiguracyjnych kompilacji jądra. Powiedział:

„Największy problem z pojedynczym sterownikiem pracującym na różnych magistralach (I2C z SPI albo I2C z I3C) jest obsługa różnych kombinacji konfiguracji (na przykład I2C=m, SPI=y).

Najprostszym rozwiązaniem jest funkcja *module_init*, która rejestrowałaby oba sterowniki, ale wymaga do wprowadzenia zależności od Kconfig w obu podsystemach. Dodatkowo nie możemy korzystać z module_i2c_driver().

Innym rozwiązaniem jest zdefiniowanie kilku *#ifdef* wraz ze skomplikowanymi zależnościami sterowników w Kconfig, tak aby rejestrować tylko sterowniki obiektów na magistralach, które są dostępne. Da się to zrobić, ale praktycznie nikt bez głębszej analizy nie zrozumie tego za pierwszym razem.

Rozwiązaliśmy to w ten sposób, że podstawowy sterownik znajduje się w jednym module, a dodatkowe jego części, dla konkretnych magistrali, znajdują się w innych modułach, dla których ponownie możemy skorzystać z *module_i2c_driver*. W jądrze znajduje się wiele wystąpień sterowników łączących w sobie I2C i SPI i działa to dobrze, chociaż powiększa się koszt w porównaniu do sytuacji z jednym sterownikiem, który mógłby na przykład korzystać z *regmap* w celu określenia różnic w funkcji *probe()*. Takie rozwiązanie sprawiłoby, że wszystko byłoby w jednym miejscu".

Dla Borisa miało to sens, gdyż widział zasadność posiadania jednego podsystemu. Dalej jednak uważał, że scenariusz łączenia obu rozwiązań nie prezentuje się dobrze. Zapytał, czy „nie możemy rozwiązać tego problemu z pomocą makra *module_i3c_i2c_driver()*, które przejęłoby całą złożoność sterowników I2C/I3C?".

Wolfram, który wcześniej zgodził się na łatki Borisa, uważał, że łączenie I2C z I3C może być czymś zupełnie innym niż łączenie I2C z SPI. W tym drugim przypadku zakłada się raczej, że system może posiadać albo jedno, albo drugie, dlatego wspieranie obu obejmowało wszystkie możliwości. Jednak wątpił, czy istnieje sprzęt działający jednocześnie na I2C i I3C. A ponieważ kod I3C jest wstecznie kompatybilny z I2C, nie istnieje potrzeba ich łączenia. Użytkownik może po prostu podłączyć urządzenie I2C i będzie działać.

Boris odpowiedział, że chociaż nie zna aktualnie urządzeń, które korzystałyby z obu protokołów, to „specyfikacja jasno określa, do czego służą statyczne adresy, i jedną z możliwości ich wykorzystania jest podłączenie urządzenia I3C do magistrali I2C, na której zachowywałoby się jak urządzenie I2C". Istnieje więc prawdopodobieństwo, i Wolfram się z tym zgodził, że w przyszłości pojawią się urządzenia korzystające z obu protokołów.

Boris próbował połączyć I2C z I3C, ale miał trudności już w fazie projektowania rozwiązania. Powiedział też: „Trudno jest zaprojektować rozwiązanie dla urządzeń, których się nie posiada. Jak na razie mogę naśladować i zmieniać I2C w zależności od potrzeb użytkownika".

W innym miejscu Greg Kroah-Hartman poprosił Borisa o oddzielenie dokumentacji od łatki i opublikowanie jej oddzielnie, tak aby kod był łatwiejszy do przejrzenia. Zapoznał się jednak z całością i miał kilka uwag, na które Boris obiecał odpowiedzieć.

Greg zauważył, dzięki brakującemu typowi danych, że Boris nigdy nie testował usunięcia urządzenia I3C, po tym jak już zostało zainstalowane. Boris odpowiedział: „Masz mnie, nigdy tego nie testowałem". Pytał potem o techniczne aspekty tego typu przypadku.

W tym momencie dyskusja zeszła na bardziej specjalistyczne tematy, gdyż Greg i Boris zastanawiali się nad problemem usuwania urządzeń w momencie pracy systemu.

Ostatecznie odpowiedź na zadane początkowo pytanie nie padła, ale wydaje się oczywiste, że kod Borisa zostanie przerobiony, tak aby uwzględnić uwagi Grega i Arnda. Jest to jednak zadanie, które może być dużo bardziej skomplikowane, niż Boris przewidywał, i raczej nie ma on urządzeń potrzebnych do testów (w pewnym momencie napisał: „wszystkie testy wykonywałem na sztucznie emulowanych urządzeniach I3C"). Wygląda więc na to, że I3C zostanie prędzej czy później włączone do jądra, ale ciągle istnieją dość duże przeszkody do pokonania.

Naprawa mmap()

Dan Williams zauważył pewien błąd. Wywołanie systemowe *mmap()* nie sprawdzało nieznanych flag, co oznaczało, że współczesne, nowe *mmap*, którego zachowanie nie współgra ze starszymi systemami, nie będzie kończyć się w akceptowalny sposób. Dan chciał zaimplementować nowe wywołanie, *mmap3()*, które by sprawdzało wszystkie flagi.

Zrodziło to kilka problemów. Po pierwsze, Christoph Hellwig zauważył, że „dodawanie nowych wywołań systemowych to bolesny proces; przerobienie wszystkich rodzajów architektury trwać będzie wieki (szczególnie w przypadku wielu architektur 32-bitowych, dla których należy wziąć pod uwagę różne poziomy granularności), a potem trzeba jeszcze uwzględnić biblioteki C, nie wspominając o aplikacjach".

Christoph zasugerował skorzystanie ze stosowanego już obejścia przez *__MAP_VALID*.

Dan odpowiedział: „Zgadzam się, że w ten sposób tworzy się bałagan i opóźnienia dla architektur oraz bibliotek *libc*, ale przecież skierowane to jest do nowych aplikacji i bibliotek, które wiedzą, że należy szukać nowej flagi, więc muszą wykonać dodatkową pracę, szukając nowego wywołania systemowego".

Odnosząc się do możliwości wykorzystania *__MAP_VALID*, zauważył, że „nowa flaga *mmap* w tym wypadku musi współpracować z *MAP_SHARED*, podczas gdy *MAP_PRIVATE* jest ignorowany".

Powiedział jednak, że jeśli nie będzie to zbyt uciążliwe, może zgodzić się na takie rozwiązanie.

Christoph zaczął sprawdzać kod *mmap* i stwierdził, że znalazł rozwiązanie problemu ignorowania *MAP_PRIVATE*. Stwierdził, że to zdecydowanie lepsza opcja niż dodawanie nowego wywołania systemowego.

Jednak Kirill A. Shutemov przyjrzał się odkryciu Christopha i zauważył, że

niektóre architektury, w szczególności PA-RISC, nie będą współpracować z tym rozwiązaniem. Christoph odpowiedział: „Nie powinniśmy się przejmować pamięcią trwałą w PA-RISC. Musimy jedynie znaleźć sposób wykluczenia PA-RISC, tak aby nie skomplikować sobie życia". Kirill jest przeciwny takiemu rozwiązaniu. Twierdzi, że interfejs wywołania systemowego powinien być uniwersalny i nie może zachowywać się inaczej w zależności od urządzenia, na którym był uruchomiony.

Dyskusja trwała, choć wydawało się, że powoli odchodzi od pierwotnego tematu. W końcu Dan wrócił do niego, mówiąc:

„Problemem jest to, że aby wspierać nowe flagi mmap, dla danych architektur należy wyszukać flagi, które na pewno nie będą działać na starszych jądrach. Przypisanie na PA-RISC wartości *0x8* do *MAP_DIRECT* nie działa, ponieważ będzie ignorowane na starszych jądrach.

Jednak już teraz niektóre architektury posiadają własne punkty dostępowe *sys_mmap*. Architektury, które nie mogą działać standardowo (wydaje się, że tylko PA-RISC), będą zmuszone dodać nowe wywołanie *mmap*. Jest to dobry kompromis pozwalający każdej innej architekturze dodać taką funkcjonalność w istniejącym wywołaniu *mmap*".

Helge Deller nie zgodził się z tym, mówiąc: „Nie chcę, aby inne architektury cierpiały z powodu PA-RISC. Jednak dodawanie nowego wywołania tylko na potrzeby PA-RISC też nie jest dobrym rozwiązaniem, ponieważ nikt potem tego u siebie nie wprowadzi".

Helge zaproponował w tym wypadku ingerencję w ABI (interfejs binarny aplikacji) dla PA-RISC. W ten sposób uzyska się prawidłowe działanie bez większych kombinacji. Dodał, że i tak nie ma zbyt wielu użytkowników PA-RISC, a większość z nich regularnie aktualizuje jądro, ponieważ w ostatnim czasie dodano wiele zmian w kodzie.

Jednak Dan odpowiedział: „Problem polega na tym, że chcemy uniknąć w ABI błędów, a w szczególności błędnych wyników przekazywanych jako prawidłowe. Utkniemy, jeśli niektóre aplikacje będą oczekiwały, że wartość *0x8* jest ignorowana, lub na odwrót, aplikacja, która musi opierać się na semantyce *MAP_SYNC*/*MAP_DIRECT*, zakłada, że wynik jest błędny, podczas gdy flagi te są ignorowane".

W tym momencie dyskusja wygasła i nie jest do końca jasne, jakie rozwiązanie zostanie przyjęte dla *mmap()*. Uważam, że obserwowanie, jak każda z propozycji trafia na przeszkody, jest niezwykle ciekawe. Wprowadzanie nowych wywołań systemowych jest czasochłonne. Proponowane obejście problemu działać będzie tylko w niektórych przypadkach. Jeśli jednak je poprawimy tak, że będzie działać zawsze, ciągle pozostaje jeden rodzaj architektury, na którym odnotujemy błędy. Z kolei ingerencja w interfejs binarny aplikacji nie jest rozwiązaniem, ponieważ właśnie tego chcieliśmy uniknąć na początku.

Oczywiście, w wielu częściach jądra przydałyby się możliwości zmian w ABI. Jednak jest to akurat coś, co społeczność chce pozostawić nienaruszone. Jedyna rzecz, jaka może wymusić zmiany w tej części jądra, to dziury w bezpieczeństwie. Bezpieczeństwo jest najważniejsze. Czyż jednak nie byłoby pięknie, gdybyśmy kiedyś dostali specjalne wydanie jądra z możliwością modyfikacji ABI, gdzie każdy fragment jądra mógłby go zmienić chociaż jeden raz? Co za szaleństwo! Co za rozpusta! A potem... poczucie winy, wzajemne oskarżenia. Niesamowite.

Śledzenie zajętości RAM-u w przypadku braku pamięci

Yang Shi zirytował się, kiedy na jego maszynie pamięć była zajęta, a OOM (out-of-memory) killer nie mógł znaleźć żadnego procesu, który mógłby zakończyć, aby uniknąć paniki jądra. Jeśli nie ma żadnego procesu do zakończenia, to dlaczego brakuje pamięci? Okazało się, że cała pamięć w systemie alokowana jest do płyt (*slab*), które mają status nieodzyskiwalnych. Yang opublikował łatkę dodającą do programu *slabinfo* opcję *-U*, dzięki której otrzymamy informacje jedynie o nieodzyskiwalnych płytach. W ten sposób użytkownik będzie mógł zidentyfikować problem i znaleźć sposób na naprawę sytuacji.

Michal Hocko nie miał nic przeciwko wprowadzeniu łatki do jądra, ale zauważył, że kod Yanga może znacznie powiększyć raport wynikowy z OOM killera, który już teraz jest niezwykle rozwlekły. Z tego powodu proponuje, aby nowa opcja była domyślnie wyłączona.

Yang nie zgodził się z tą uwagą, mówiąc, że dodatkowe dane w raporcie pojawią się tylko w przypadku poważnych awarii, a nie jako element standardowej procedury. Dodał też, że bez problemu można dodać plik do systemu plików proc (*procfs*), dzięki czemu w każdej sytuacji możliwa będzie kontrola nad zwracanymi przez program danymi.

Jednak Michal stwierdził, że dla niego nie ma to znaczenia. Powiedział: „Większość raportów OOM, które widziałem, to była pamięć przypięta (pinned) z przestrzeni użytkownika". Woli jednak, aby raport był bardziej obszerny niż mniej.

Kontynuowali dyskusję i w końcu doszli do porozumienia, kiedy to Yang zasugerował: „Może powinniśmy podać informację o stosunku nieodzyskiwalnych płyt do całości pamięci. Na przykład kiedy osiąga on wartość równą i większą od 50%, wtedy podajemy statystyki płyt w OOM? Dodatkowo procent ten można by było zmieniać w */proc*.".

Michal zgodził się, że ma to sens, i na tym wątek się zakończył. ■■■

Tails 3.4

Na płycie DVD dołączone do bieżącego wydania „Linux Magazine” znajdziemy znakomitą dystrybucję Tails 3.4, której głównym zadaniem jest zapewnienie nam anonimowości w sieci.

W czasach mediów społecznościowych troska o prywatność użytkowników może brzmieć jak głos wołającego na puszczy. Użytkownicy sami podają na tacy firmom trzecim najbardziej intymne informacje na swój temat – czy to w formie e-maili przesyłanych darmową pocztą, czy wiadomości wysyłanych popularnymi komunikatorami, aktualizacji statusu, oznaczeń na zdjęciach itd. Firmy takie jak Google czy Facebook często wiedzą o niektórych osobach znacznie więcej, niż ich rodziny. Jak pokazuje praktyka, pracownicy tych firm nie zawsze potrafią oprzeć się pokusie i nie korzystać z tej wiedzy. Niezależnie od tego jesteśmy cały czas profilowani, by klienci wspomnianych firm mieli jak największe szanse zdobyć to, co dla nas najcenniejsze: naszą uwagę. Im bardziej kompletny jest nasz profil, tym większe szanse na zwrócenie uwagi, a co za tym idzie – sprzedaż produktu lub usługi.

Nie wszystkim się to podoba. Są osoby, które nie życzą sobie, by być na każdym kroku śledzonym i profilowanym. Wydaje się jednak, że nie mamy podstawowego prawa do prywatności w Internecie – musimy je sobie wywalczyć. Jednym z narzędzi, które mogą nam w tym pomóc, jest Tails. To kompletny system uruchamiany z płyty DVD, dzięki któremu pozostaniemy całkowicie anonimowi – oczywiście dopóty, dopóki sami nie zdecydujemy się na ujawnienie swojej tożsamości, np. przez użycie takiego samego identyfikatora konta na forum internetowym co w serwisie aukcyjnym (do którego przypisane są rzeczywiste dane).

Do zapewnienia anonimowości Tails wykorzystuje sieć Tor, która oparta jest na tzw. trasowaniu cebulowym. Dzięki niemu ruch sieciowy między naszym komputerem a serwerem docelowym przechodzi przez wiele maszyn w taki sposób, że nie sposób określić IP systemu źródłowego. Jednak samo użycie Tora jest słabą ochroną, istnieje bowiem wiele sposobów identyfikacji systemu. Aby używać Tora we właściwy sposób, potrzebujemy pełnego systemu, który zagwarantuje, że nie zdradzi nas żaden komponent systemu. W przypadku zamkniętych systemów jest to bardzo trudne, ponieważ nie sprawujemy nad nimi pełnej kontroli. Linux jest znacznie lepszym rozwiązaniem – jednak zamiast systemu ogólnego zastosowania lepiej sprawdzi się dystrybucja specjalnie przystosowana do tego celu. Taką właśnie dystrybucją jest Tails.

Co ciekawe, w 2014 r. dziennikarze ujawnili fragment kodu źródłowego narzędzia Xkeyscore. Program ten służy do analizy przechwytywanego ruchu sieciowego: decyduje, które dane zignorować, a które zachować. Okazało się, że na szczególną uwagę NSA zasłużył nie tylko Tor, ale i Tails, a nawet witryna Linux Journal, która zamieściła opis dystrybucji. Oznacza to, że nawet zwykli użytkownicy niezainteresowani Torem, którzy odwiedzili witrynę czasopisma linuksowego, zostali uznani za wartych zainteresowania. [1]

Jak korzystać z Tailsa? Umieszczamy płytę DVD w napędzie i restartujemy komputer. System powinien uruchomić się z płyty – jeśli tak się nie stanie, powinniśmy zmienić odpowiednią opcję w BIOS-ie. Po uruchomieniu zobaczymy ekran powitalny Tailsa, na którym znajdziemy konfigurator ustawień regionalnych (język, klawiatura, jednostki) i opcjonalnie wprowadzić hasło pozwalające odszyfrować pamięć masową, jeśli chcemy trwale przechowywać jakieś dane. Następnie naciskamy przycisk „Start Tails”. Odtąd możemy korzystać z przeglądarki i innych narzędzi znajdujących się w systemie, pamiętając, że jeżeli faktycznie zależy nam na zachowaniu anonimowości, powinniśmy bardzo uważać na każdy swój krok. ■■■

INFO

[1] NSA interesuje się osobami odwiedzającymi witrynę czasopisma linuksowego: http://www.linuxjournal.com/content/nsa-linux-journal-extremist-forum-and-its-readers-get-flagged-extra-surveillance

PRENUMERATA i E-PRENUMERATA
LINUX MAGAZINE

DVD Z NAJNOWSZYMI DYSTRYBUCJAMI LINUKSA*

Ubuntu • Fedora • Linux Mint • openSUSE • Knoppix • CentOS • Mageia

PRENUMERATA E-PRENUMERATA

zamów: prenumerata@linux-magazine.pl, 22 429 43 05

prywatność • sieci • chmury • duże dane • mobilność • odzyskiwanie danych • bezpieczeństwo
wirtualizacja • administracja systemem • wydajność • optymalizacja • interoperacyjność
lekcje bezpieczeństwa • warsztat admina • wiersz poleceń • przestrzeń robocza
Klaus Knopper • Jon "maddog" Hall

6 NUMERÓW W CENIE 4!
12 NUMERÓW W CENIE 8!
24 NUMERY W CENIE 13!
NISKIE CENY E-WYDAŃ

SZCZEGÓŁY:

http://linuxmagazine.pl/index.php/subscribe
http://linuxmagazine.pl/index.php/eprenumerata

* w prenumeracie wydań drukowanych

Personalizacja rozruchu UEFI

Sprytny **start**

Tradycyjny BIOS liczy już sobie dziesięciolecia i nie nadąża za szybkim rozwojem pecetów i laptopów. Jego zadania przejmuje potężny następca, UEFI, i może pochwalić się większą funkcjonalnością, wygodą i bezpieczeństwem. Maik Brüggemann i Ralf Spenneberg

Już od 1981 roku pecety uruchamiają się na bazie BIOS-u – „podstawowego systemu wejść i wyjść" (Basic Input Output System). Na przestrzeni lat różni producenci próbowali go rozszerzyć, ale dziś ten system oprogramowania układowego jest trudny w dopasowaniu i wciąż nie działa na architekturze 64-bitowej.

Pod koniec 1992 roku Intel ruszył z inicjatywą zastąpienia sędziwego BIOS-u 64-bitową alternatywą. Intelowska „inicjatywa na rzecz rozszerzalnego interfejsu oprogramowania układowego" (Extensible Firmware Interface Initiative, EFI) w 2005 roku rozwinęła się do „forum zjednoczonego EFI" (UEFI Forum). Dziś oprócz Intela w projekt angażuje się AMD, Microsoft, HP i inni najwięksi producenci sprzętu.

UEFI to opis interfejsu pomiędzy oprogramowaniem układowym komputera a systemem operacyjnym. Podobnie jak BIOS, implementacja UEFI inicjalizuje podzespoły, aby mógł wystartować system operacyjny. Różnica jest taka, że UEFI natywnie obsługuje architekturę 64-bitową i interfejsy graficzne oraz może nas chronić przed złośliwym oprogramowaniem, akceptując tylko podpisane systemy operacyjne (ta ostatnia funkcja znana jest jako Secure Boot).

Jeśli korzystamy z Linuksa, który od razu obsługuje UEFI i Secure Boot, nie musimy łamać sobie głowy nad procesem rozruchu, chyba że chcemy go spersonalizować albo naprawić. Ciekawskim UEFI może za to pokazać swoją wszechstronność i bogate możliwości. W tym artykule przyjrzymy się rozruchowi w UEFI i wykorzystaniu tej platformy do stworzenia aplikacji niezależnej od systemu operacyjnego.

Fazy rozruchu

Oprogramowanie układowe UEFI jest modułowe i rozszerzalne. W momencie załączenia zasilania komputera oprogramowanie układowe przechodzi przez wiele etapów w ściśle określonej kolejności (Rysunek 1). Faza pierwsza to Security Phase (SEC), ale to dość niefortunna nazwa, ponieważ w kwestii bezpieczeństwa niewiele się wtedy dzieje – to nie podczas tego etapu oprogramowanie układowe UEFI sprawdza podpisy. Najogólniej rzecz ujmując, w tej fazie inicjalizuje się procesor i jego kod sprzętowy, ładowany bezpośrednio z pamięci flash, ponieważ pamięć RAM jeszcze nie jest dostępna. Oprogramowanie UEFI w roli RAM-u wykorzystuje tymczasową pamięć cache procesora. Dane o rozmiarze i położeniu tego tymczasowego magazynu oraz inne opcjonalne informacje o procesorze przechodzą z fazy SEC do fazy pre-EFI.

W trakcie inicjalizacji pre-EFI (PEI) staje się dostępna właściwa pamięć operacyjna oraz podzespoły niezbędne do kolejnych faz. Wymiana między podzespołami następuje dzięki modułom inicjalizacyjnym pre-EFI (PEIM) zapewniającym im API o nazwie PEIM-to-PEIM interface (PPI). Dyspozytor modułów PEIM odpowiada za ich załadowanie i działanie: sprawdza zależności między nimi, ładuje je do dostępnego już RAM-u i uruchamia w określonej kolejności.

Oprogramowanie układowe UEFI nie tylko inicjalizuje podstawowe podzespoły. Odczytuje również uporządkowany zbiór kodu i danych z lokalizacji zwanej woluminem dla oprogramowania układowego (*firmware volume*). Te dane przechowywane są zwykle w pamięci flash i zawierają sterowniki urządzeń potrzebnych w kolejnej fazie. Gdy dyspozytor uruchomi już większość potrzebnych PEIM, na koniec poszukuje PEIM o nazwie DXE IPL, który rozpoczyna kolejną fazę rozruchu.

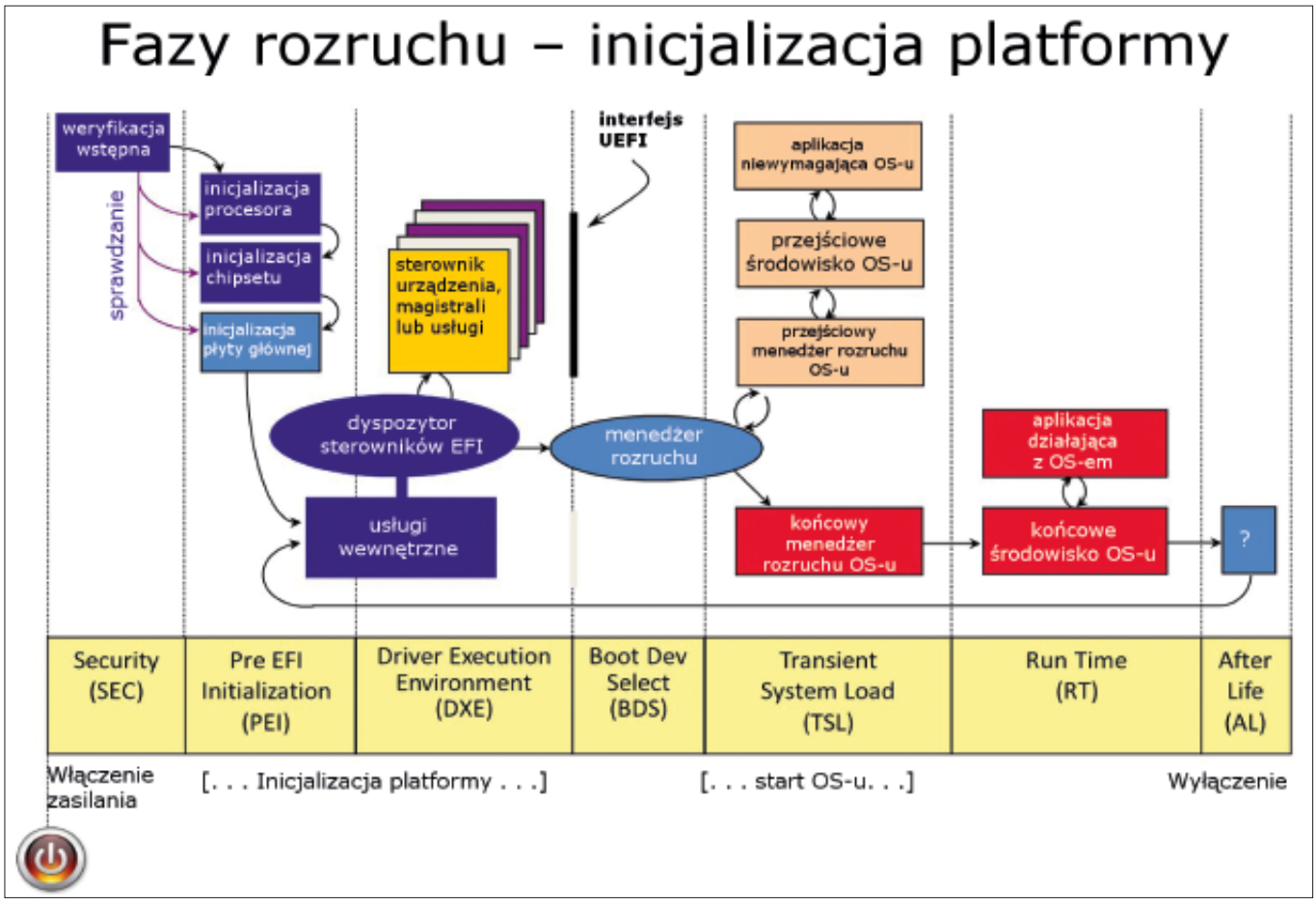
W fazie DXE, czyli w „środowisku wykonywalnym sterowników", odbywa się większa część niezbędnej inicjalizacji komputera. Podobnie jak w fazie PEI, w fazie DXE również występuje dyspozytor ładujący sterowniki DXE z *firmware volume* i uruchamia w ustalonej kolejności. Sterowniki inicjalizują podzespoły i rejestrują lub wykorzystują protokoły.

Właśnie przez protokoły na konsolę trafiają dane tekstowe i możliwy jest dostęp do urządzeń PCI. Protokół dostępny tylko przed startem systemu operacyjnego nosi nazwę usługi rozruchowej (*boot service*), natomiast ten dostępny w trakcie pracy systemu operacyjnego to usługa uruchomieniowa (*runtime service*).

Kiedy dyspozytor załaduje wszystkie sterowniki, następuje faza wyboru urządzenia rozruchowego (BDS) uruchamiająca menedżer rozruchu UEFI. Zmienne z pamięci NVRAM informują menedżera rozruchu, które aplikacje UEFI należy uruchomić. Na te zmienne poprzez usługę uruchomieniową może wpływać system operacyjny, a tym samym wyznaczać aplikacje do uruchomienia w tym momencie. Aplikacje UEFI, na przykład do diagnostyki czy odzyskiwania danych rzeczywiście bywają często oferowane przez producentów sprzętu. Do UEFI można też dodać własne aplikacje, aby stanowiły część procesu uruchomieniowego.

Szczególną aplikacją jest menedżer rozruchu systemu operacyjnego, czyli bootloader. Komputer zwykle ładuje i uruchamia ją z partycji systemowej EFI (*/boot/efi*). Po uruchomieniu bootloadera rozruch przechodzi do fazy przejściowego ładowania systemu (TSL). Jeśli komputer korzysta z technologii Secure Boot, na tym etapie jest sprawdzany podpis systemu operacyjnego, a kod bez ważnego podpisu się nie uruchomi. Faza TSL kończy się wywołaniem funkcji *ExitBootServices()*, która dokładnie czyści pamięć, pozostawiając w niej tylko specjalnie oznaczone interfejsy potrzebne działającemu systemowi operacyjnemu.

Po wywołaniu *ExitBootServices()* komputer znajduje się w fazie uruchomienia (*runtime*, RT): system operacyjny działa i ma dostęp do usług uruchomieniowych UEFI. Po fazie uruchomienia komputer może wejść w fazę *After-Life* (AL), by zakończyć



Rysunek 1: Proces rozruchu UEFI ma kilka faz.

działanie systemu operacyjnego. Tę fazę zaprojektowano pod kątem czystego wyłączania komputera, ale jest rzadko używana.

Usługi i zmienne uruchomieniowe UEFI

Usługi uruchomieniowe UEFI to takie jego funkcje, z których można korzystać w trakcie pracy systemu operacyjnego. Jeden ze sterowników UEFI rejestruje się jako usługa uruchomieniowa i tworzy wpis w tabeli takich usług. W procesie rozruchu system operacyjny uzyskuje dostęp do tej tabeli, dzięki czemu ma dostęp do zegara czasu rzeczywistego – może spowodować ponowny rozruch całej platformy, dostać się do zmiennych w pamięci NVRAM lub zainicjować aktualizację oprogramowania układowego.

Na maszynie z Linuksem większość usług uruchomieniowych dostępna jest tylko dla sterowników jądra, ale dostęp do zmiennych pamięci NVRAM i funkcja aktualizacji oprogramowania układowego są przez jądro eksportowane do przestrzeni użytkownika. Dzieje się to za pośrednictwem systemu plików dla zmiennych UEFI (*UEFI variable filesystem, efivarfs*). Dystrybucje Linuksa uruchomione przez UEFI włączają taki system plików do swego drzewa pod lokalizacją */sys/firmware/efi/efivars*.

Listę wszystkich dostępnych zmiennych uzyskamy zwykłym poleceniem *ls /sys/firmware/efi/efivars*, a ich zawartość można odczytywać jak zawartość zwykłych plików – wypiszemy ją na terminal poleceniem:

```
cat /sys/firmware/efi/
efivars/variable_name
```

Problem w tym, że dane przechowywane w zmiennych znajdują się tam w postaci maszynowej, a wynik polecenia *cat* często wygląda bardzo tajemniczo.

Do zmiennych można coś wpisywać lub kasować je jak zwykłe pliki, ale to drugie jest szczególnie niebezpieczne. Do niedawna polecenie *rm-rf /* skutkowało usunięciem z dysku twardego nie tylko wszystkich plików, ale także wszystkich zmiennych UEFI. Niektóre posiadające błędy implementacje UEFI nadal spodziewały się konkretnych zmiennych, więc jeśli te były usunięte, to komputer nie uruchamiał się, a płyta główna trafiała do naprawy.

Z powodu wspomnianych wyżej sytuacji w nowszych jądrach Linuksa większość zmiennych UEFI jest chroniona nienaruszalną flagą:

```
lsattr PKDefault-8be4df61-93
ca-11d2-aa0d-00e098032b8c
----i----------
PKDefault-8be4df61-93ca
-11d2-aa0d-00e098032b8
```

LISTING 1: Konfiguracja fazy BDS

```
# efibootmgr --verbose
BootCurrent: 0003
Timeout: 1 seconds
BootOrder: 0003.0002
Boot0002 Hard Drive BBS(HD,,0x0)
Boot0003* debian HD(1, GPT, cf2d93bb-3cd3-4903-9b7ccbfc-
697f7aae,  0x800, 0x1dc800)/File(\EFI\debian\grubx64.efi)..BO
```

Tym sposobem wykluczono przypadkową zmianę lub usunięcie zmiennych. W praktyce nie ma bezpośredniego dostępu do zmiennych przez polecenia takie jak *cat* i do edycji grupy zmiennych rozruchowych należy użyć specjalnych programów, na przykład *efibootmgr*.

Zmienne rozruchowe mają wpływ na przebieg fazy wyboru urządzenia rozruchowego (BDS). Zmienna zaczynająca się na *Boot000* stanowi wpis do menedżera rozruchu UEFI – jej zawartość określa położenie aplikacji UEFI przeznaczonej do uruchomienia w fazie BDS. Zmienna *BootOrder-GUID* określa natomiast kolejność uruchamiania aplikacji. Tymi zmiennymi można manipulować za pomocą konsolowego programu *efibootmgr*. Polecenie z Listingu 1 wyświetli nam aktualną konfigurację fazy BDS.

W Listingu 1 mamy dwa wpisy dotyczące rozruchu. Wpis *debian* jest na samym początku sekwencji, więc jest przetwarzany jako pierwszy. Wskazuje na partycję systemową EFI (*/boot/efi*), na której znajduje się Grub. Komputer w fazie BDS uruchamia najpierw implementację Gruba na UEFI, Grub uruchamia zaś jądro Linuksa. Nowe zmienne rozruchowe lub inną kolejność możemy ustawić poleceniem *efibootmgr*.

O ile zmienne rozruchowe można dowolnie dostosowywać, o tyle dostęp do innych zmiennych, na przykład *PK*, *KEK*, *DB* i *DBX*, jest już ograniczony. Te zmienne przechowują klucze publiczne do weryfikacji podpisu przy bezpiecznym rozruchu.

Jeśli atakujący jest w stanie wpłynąć na zawartość tych zmiennych, to może również obejść lub wyłączyć mechanizm Secure Boot, dlatego mogą tam być przechowywane tylko dane z podpisem. Implementacja UEFI sprawdza podpis przekazywanego klucza publicznego i pozwala na zapis, jeśli podpis przejdzie weryfikację.

Kapsuła aktualizacji

Kolejna ciekawa usługa uruchomieniowa UEFI nosi nazwę kapsuły aktualizacji (*Update Capsule*), za pomocą której system operacyjny przekazuje bloki danych do oprogramowania układowego UEFI. Za pomocą tej usługi system operacyjny zapisuje jakieś dane w pamięci RAM i informuje firmware o ich położeniu, po czym następuje reset systemu i restart komputera, a UEFI może dostać się do wskazanego bloku danych.

Tę usługę wykorzystuje się głównie do przeprowadzania aktualizacji oprogramowania układowego UEFI, które de facto wykonuje tę operację samo na sobie. Jest to duża zaleta, ponieważ nie potrzebujemy żadnych własnościowych narzędzi, zwykle dostępnych tylko dla systemu Windows.

Standardowy interfejs UEFI uniezależnia aktualizacje oprogramowania układowego od systemu operacyjnego. Microsoft korzysta z procesu aktualizacji w taki sposób, by Windows Update wgrywał nowe oprogramowanie układowe na tablety Surface, a Apple implementuje aktualizację przez UEFI na niektórych MacBookach. Użytkownicy Linuksa mogą obecnie użyć tej funkcji aktualizacji tylko na niektórych komputerach. Poza Microsoftem i Apple jedynym dużym producentem sprzętu wspierającym aktualizacje przez kapsułę jest Dell.

Podobnie jak zmienne UEFI, również interfejs aktualizacji jest eksportowany przez jądro Linuksa do systemu plików. Lokalizacja */sys/firmware/efi/esrt* zawiera tablicę zasobów systemowych EFI (ESRT). Jądro generuje dane wejściowe dla każdego podzespołu, który można zaktualizować przez interfejs

kapsuły. Listę wszystkich urządzeń uzyskamy z narzędzia *fwupdate*, instalowanego poleceniem:

```
sudo apt-get install fwupdate
```

Wyświetlamy listę urządzeń współpracujących z kapsułą aktualizacji:

```
fwupdate --list
```

Jeśli dla danego urządzenia z listy jest dostępne oprogramowanie układowe, instalujemy je tak:

```
fwupdate --apply=
<GUID-sprzętu> <plik-z-oprogramowaniem>.cap
```

Jak już wspomniano, tę funkcjonalność wspiera na razie tylko kilku producentów, jednak jest ona dobrym przykładem przewagi interfejsu UEFI nad własnościowymi implementacjami BIOS-u.

Własny kod

Aplikacje na UEFI tworzy się w ramach zestawu deweloperskiego EFI Developer Kit II (EDK II). Obecnie jest to wzorcowa implementacja specyfikacji UEFI, używana przez wielu producentów jako fundament ich oprogramowania układowego. Aby deweloperzy nie musieli ciągle restartować komputerów, by sprawdzić działanie aplikacji, z zestawem najlepiej jest pracować w Qemu. EDK II zawiera wszystkie elementy potrzebne do wygenerowania kompletnego programu układowego UEFI i zasymulowania rozruchu pod Qemu.

Zaczynamy od zainstalowania zależności. W Ubuntu 16.04 będzie to polecenie:

```
sudo apt-get install
build-essential uuid-dev
iasl git gcc-5 nasm
```

Najnowsze stabilne wydanie EDK II najłatwiej pobrać z GitHuba:

```
mkdir ~/src
cd ~/src
git clone https://github.com/
tianocore/edk2.git vUDK2017
```

Po ściągnięciu EDK kompilujemy BaseTools:

```
cd ~/src/vUDK2017
make -C BaseTools
```

Jeśli wszystko odbyło się poprawnie, można już korzystać z EDK. Kod źródłowy jest poukładany w pakiety, z których najważniejsze to *MdeModulePkg* i *MdePkg*, implementujące specyfikację UEFI jako taką. Kod EDK jest w ogromnej większości niezależny od architektury – aby było możliwe wygenerowanie programu układowego na konkretną platformę, należy dodać odpowiedzialny za nią kod. W ramach EDK otrzymujemy taką kompatybilność z Qemu. Pakiet Ovmf opiera się na pakietach Mde i rozszerza środowisko o elementy charakterystyczne dla Qemu.

Aby pakiet Ovmf utworzył nowe EDK, edytujemy plik *~/src/vUDK2017/Conf/ target.txt*, a jako *ACTIVE_PLATFORM* podajemy pakiet Omvf:

```
ACTIVE_PLATFORM =
OvmfPkg/OvmfPkgX64.dsc
```

Ponadto trzeba odpowiednio dopasować używaną architekturę i łańcuch narzędzi kompilatora. W systemie Ubuntu 16.04 x64 będą to wartości:

```
TARGET_ARCH= X64
TOOL_CHAIN_TAG= GCC5
```

Docelowa platforma jest skonfigurowana, więc uruchamiamy pierwszą kompilację:

```
source edksetup.sh
build
```

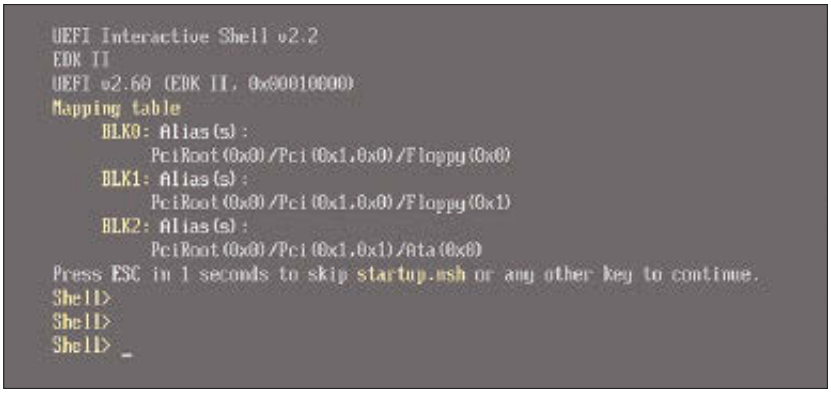
Budowanie programu układowego zajmuje kilka minut. Gotowy plik znajdziemy w *~/src/vUDK2017/Build/OvmfX64/DEBUG_GCC5/FV*

Aby przetestować nasz program układowy UEFI pod Qemu, kopiujemy jego pliki do nowego katalogu (Listing 2). Kolejnym poleceniem uruchamiamy maszynę wirtualną, a na niej program:

```
Qemu-system-x86_64 -drive
if=pflash,format=raw,
file=OVMF_CODE.fd -drive
if=pflash,format=raw,file=OVMF_VARS.fd
```

Listing 2: Kopiowanie pliku z programem układowym

```
01 mkdir ~/efibin
02 cd ~/efibin
03 cp ~/src/vUDK2017/Build/OvmfX64/DEBUG_GCC5/FV/OVMF_CODE.fd .
04 cp ~/src/vUDK2017/Build/OvmfX64/DEBUG_GCC5/FV/OVMF_VARS.fd .
```



Rysunek 2: Uruchomiona powłoka UEFI.

Listing 3: Witaj, świecie! – przykładowy program UEFI

```
01 #include <Uefi.h>
02 #include <Library/PcdLib.h>
03 #include <Library/UefiLib.h>
04 #include <Library/UefiApplicationEntryPoint.h>
05
06 EFI_STATUS
07 EFIAPI
08 UefiMain (
09  IN EFI_HANDLEImageHandle,
10  IN EFI_SYSTEM_TABLE *SystemTable
11  )
12 {
13      UINTN Index;
14      EFI_INPUT_KEY Key;
15      CHAR16 *result = L"pingwin";
16      CHAR16 *state = L"_______";
17      UINTN Attempt = 0;
18
19      do {
20           Attempt++;
21           SystemTable->ConOut->ClearScreen(SystemTable->ConOut);
22
23           Print(L"Witamy w grze Wisielec\n");
24           Print(L"Próba nr: %d\n\n\n", Attempt);
25
26           // wypisuj znalezione znaki
27           for(UINTN i=0; i<StrLen(state); i++) {
28                Print(L"%c ", state[i]);
29           }
30
31           // odczytuj znaki z klawiatury
32           SystemTable->BootServices->WaitForEvent(1, &SystemTable-
>ConIn->WaitForKey, &Index);
33           SystemTable->ConIn->ReadKeyStroke(SystemTable->ConIn,
&Key);
34
35           // sprawdzaj, czy znak z wciskanego klawisza jest w haśle
36           for(UINTN i=0; i<StrLen(result); i++) {
37                if (Key.UnicodeChar == result[i]) {
38                     state[i] = result[i];
39                }
40           }
41      } while(StrCmp(result, state) != 0); // powtarzaj do odgadnięcia
wszystkich znaków
42
43      // wypisz wynik
44      SystemTable->ConOut->SetCursorPosition(SystemTable-
>ConOut, 0 , 4);
45      for(UINTN i=0; i<StrLen(state); i++) {
46                Print(L"%c ", state[i]);
47      }
48      // czekaj na klawisz
49      Print(L"\n\nGratulacje! Naciśnij dowolny klawisz, aby
kontynuować\n");
50      SystemTable->BootServices->WaitForEvent (1, &SystemTable-
>ConIn->WaitForKey,
51      &Index);
52
53      return EFI_SUCCESS;
54 }
```

Jeśli Qemu krótko wyświetli logo z wyrazami *TianoCore*, a następnie uruchomi powłokę UEFI (Rysunek 2), to znaczy, że operacja przebiegła pomyślnie.

Do pierwszych prób z aplikacjami UEFI można użyć na wzór programu *Hello world* z *MdeModule* – jego położenie to *MdeModulePkg/Application/HelloWorld/HelloWorld.c*. Plik, którego treść widzimy w Listingu 3, rozszerzono jeszcze o implementację prostej gry w wisielca. Gdy użytkownik oddadnie słowo *pingwin*, aplikacja kończy działanie.

Punktem wejścia do aplikacji jest funkcja *UefiMain* (wiersz 8), przyjmująca dwa argumenty z zewnątrz. Pierwszy, *ImageHandle*, to uchwyt samego procesu, bez znaczenia przy analizowaniu naszego programu, natomiast drugi argument, *SystemTable*, to centralna struktura danych w środowisku UEFI. W tym momencie daje ona aplikacji dostęp do protokołów zarejestrowanych wcześniej w fazie DXE.

W wierszu 21 program korzysta przez tablicę systemową z prostego protokołu wypisywania tekstu (*Simple Text Output Protocol*). W ramach protokołu działa funkcja *ClearScreen*. Jak wskazuje nazwa, wywołanie jej powoduje wyczyszczenie konsoli wypisującej dane. Następnie dochodzą nowe dane z funkcji wypisującej, która również ma pod maską protokół wypisywania tekstu. W linii 32 program korzysta z usługi *WaitForEvent*, czyli czeka na wystąpienie określonych zdarzeń – tu na wciśnięcie klawisza (*WaitForKey*). Po wystąpieniu zdarzenia funkcja *ReadKeyStroke* zwraca wartość wciśniętego klawisza.

Następnie pętla *for* sprawdza, czy wprowadzony znak pasuje do któregoś z oczekiwanych znaków. Cały proces powtarza się w pętli *do-while*, aż użytkownik odgadnie wszystkie literki. Kiedy to nastąpi, program wypisuje odgadnięte słowo i czeka na wciśnięcie klawisza, po czym kończy działanie.

Aby program z gierką znalazł się w następnej kompilacji, trzeba rozszerzyć pakiet Ovmf. W pliku *~/src/vUDK2017/OvmfPkg/OvmfPkgX64.dsc* dopiszmy część *[Components]* i odnośnik do aplikacji Hello World:

```
[Components]
MdeModulePkg/Application/
HelloWorld/HelloWorld.inf
```

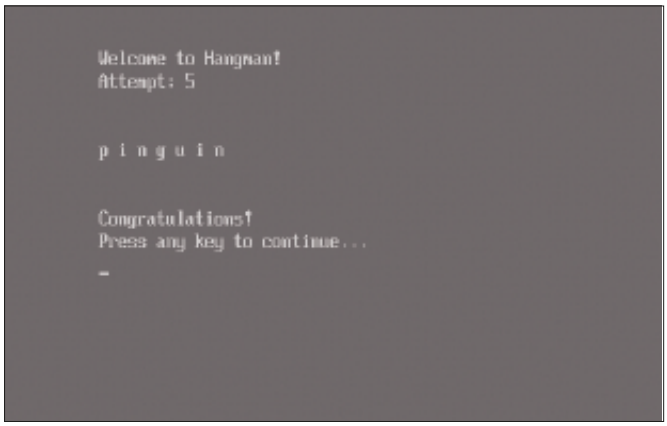
Plik *HelloWorld.inf* jest opisem aplikacji i instrukcją dla EDK, aby automatycznie generował pliki Makefile. W głównym folderze EDK uruchamiamy kompilację programu Hello World:

```
source edksetup.sh
build
```

Wygenerowana biblioteka UEFI *HelloWorld.efi* trafiła do lokalizacji *~/src/Build/OvmfX64/DEBUG_GCC5/X64*. Aby przetestować aplikację, kopiujemy ją do katalogu *efibin* i ponownie uruchamiamy Qemu:

```
cd ~/efibin
cp ~/src/Build/OvmfX64/
DEBUG_GCC5/X64/HelloWorld.efi .
```

Polecenie uruchamiające system Qemu rozszerzamy o parametr *hda*, który włącza katalog *efibin* do maszyny wirtualnej jako dysk twardy FAT. Pełne polecenie wygląda tak:



Rysunek 3: Gierka w wisielca sterowana z UEFI.

```
Qemu-system-x86_64 -drive
if=pflash,format=raw,
file=OVMF_CODE.fd -drive
if=pflash,format=raw,
file=OVMF_VARS.fd -hda fat:~/efibin
```

UEFI bezpośrednio wspiera system plików FAT, więc będzie miało dostęp do aplikacji *Hello World*. Gdy uruchomi się powłoka UEFI, program wywołujemy, wpisując *Hello World* (Rysunek 3).

Gdy skończymy pisać własną aplikację na UEFI, gotowy plik binarny możemy uruchamiać w środowisku UEFI na płycie głównej. Przekopiujmy go na partycję systemową EFI w komputerze:

```
sudo cp ~/efibin/HelloWorld.efi
/boot/efi/
```

Podobnie jak w Qemu, aplikację w UEFI na płycie głównej można uruchomić ręcznie z powłoki UEFI. Po restarcie komputera wciskamy jeden z klawiszy funkcyjnych, aby przerwać zwykły proces rozruchowy i wejść do powłoki UEFI (klawisz jest inny w zależności od implementacji UEFI na płycie). Kiedy już jesteśmy w powłoce, wywołujemy aplikację, wpisując *HelloWorld*.

Innym rozwiązaniem jest ustawienie automatycznego uruchamiania aplikacji przez konfigurację menedżera rozruchu UEFI za pomocą zmiennych. Najłatwiej zrobić to w programie *efibootmgr*, który generuje nowe zmienne rozruchowe w systemie plików *efivar*, a w menedżerze rozruchu pojawia się nowy wpis.

Przed wykonaniem tego kroku lepiej upewnić się, czy mamy płytę ratunkową, z poziomu której naprawimy możliwy błąd konfiguracji. Nowy wpis w menedżerze powstaje przez przekazanie do programu Efibootmgr parametru *create*. Pod *label* ustawiamy intuicyjną etykietę wpisu, a *loader* pokazuje programowi ścieżkę do aplikacji UEFI:

```
efibootmgr --create --label
"Wisielec" --loader HelloWorld.efi
```

Jeśli operacja się udaje, system odpowiada:

```
BootCurrent: 0000
Timeout: 2 seconds
BootOrder: 0000,0001
Boot0000* debian
Boot0001* Hangman
```

W sekwencji rozruchu wpis o etykiecie *debian* nadal jest ustawiony jako pierwszy. Aby następny rozruch zaczął się od wpisu o numerze 0001, stosujemy polecenie *efibootmgr --bootnext 0001*. Kiedy teraz zrestartujemy komputer, oprogramowanie układowe UEFI jednorazowo uruchomi naszą aplikację *Hello World*.

Wnioski

Zaprezentowany przykład pisania i uruchamiania programów na UEFI jest bardzo prosty, ale nie jesteśmy tutaj ograniczeni tylko do wpisywania i wypisywania tekstu. UEFI pozwala również na tworzenie graficznych interfejsów użytkownika i dostęp do sieci TCP/IP czy pendrive'a. To wszechstronne środowisko, w którym uruchomimy nawet złożone aplikacje niezależnie od systemu operacyjnego. ■■■

Linux kontroluje bezpieczny rozruch

Lepszy start

Shim pozwala użytkownikom Linuksa odzyskać kontrolę nad procesem bezpiecznego rozruchu.

Eva-Katharina Kunst i Jürgen Quade

AUTORZY

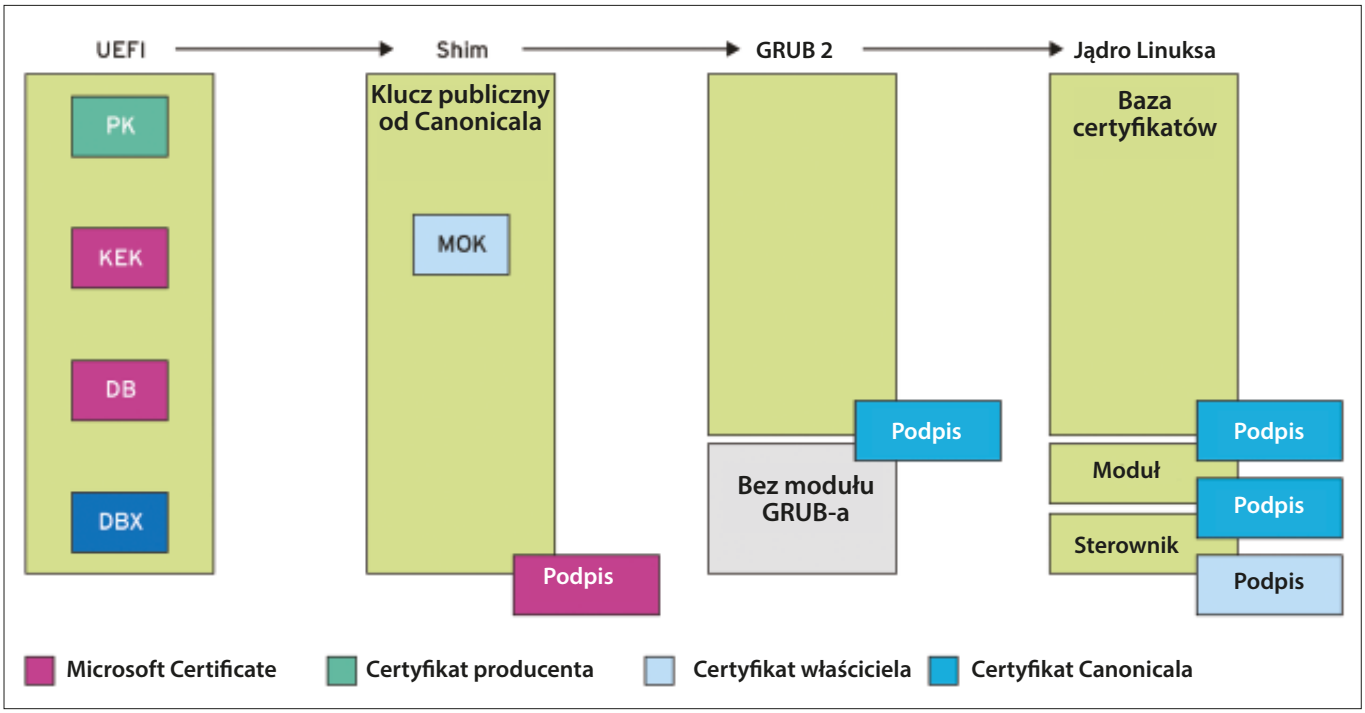
Eva-Katharina Kunst jest fanką Linuksa od czasów zaistnienia otwartego oprogramowania.

Jürgen Quade to profesor Uniwersytetu Niederrhein w Krefeld w Niemczech. Czwarta edycja ich wspólnej książki „Linux-Treiber entwickeln" [Tworzenie linuksowych sterowników] wydana została pod koniec 2015 roku.

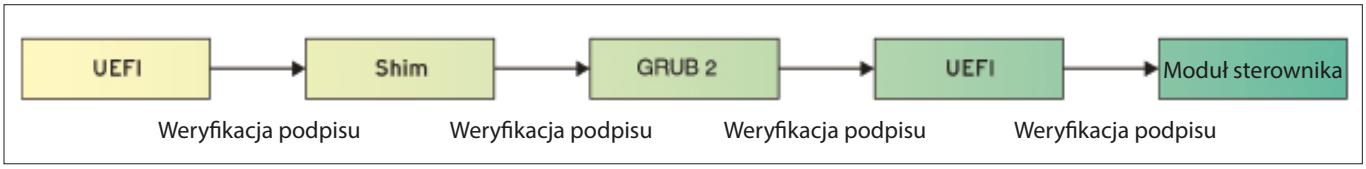
Bezpieczny rozruch UEFI sprawia, że na komputerze można uruchomić jedynie oprogramowanie z aktualnym cyfrowym podpisem. UEFI szuka na dysku programu rozruchowego, weryfikuje podpis z jednego z certyfikatów, które przechowuje i, jeśli jest on poprawny, ładuje i aktywuje kod programu.

Program rozruchowy wyszukuje system operacyjny, weryfikuje cyfrowy podpis i uruchamia system, po czym wgrywa podpisane moduły jądra i sterowniki.

Chodzi o to, że jeśli uruchamiamy jedynie kod ze sprawdzonego źródła, autorom malware'u, chowającym się w ciemnych zakamarkach Internetu, jest znacznie



Rysunek 1: Przechowywanie certyfikatów na komputerze z bezpiecznym rozruchem. DBX to baza zabronionych podpisów.



Rysunek 2: Bezpieczny rozruch w Ubuntu przechodzi przez kilka faz.

trudniej przemycić swoje oprogramowanie do procesu rozruchu.

Dla użytkowników i deweloperów Linuksa problem takiego rozwiązania polega na tym, że kontrolę nad nim sprawuje Microsoft. Firma ta ma silną pozycję na rynku, wobec czego twórcy sprzętu tworzą własne certyfikaty, nazywane Kluczami Platformy (PK, Platform Key), a następnie umieszczają certyfikaty Microsoftu są w bazach KEK (Key Exchange Key) i autoryzowanej bazie kluczy (DB) na płycie głównej (Rysunek 1). Dlatego komputery architektury x86 uruchamiają jedynie oprogramowanie, które posiada błogosławieństwo Microsoftu.

Shim: rozwiązanie alternatywne

Pomysł, żeby jądro Linuksa potrzebowało cyfrowego podpisu Microsoftu, był na tyle kontrowersyjny, że Matthew Garrett stworzył program rozruchowy nazwany Shim, który jest otwartoźródłową alternatywą zbierającą własne certyfikaty. Ubuntu, Red Hat, SUSE i Debian tworzą swoje wersje Shima zawierające certyfikaty wydane przez te firmy.

W zastępstwie Microsoftu program rozruchowy jest podpisywany przez Verisign/Symantec, dzięki czemu oprogramowanie sprzętowe UEFI może uruchomić Shima. Kiedy już to nastąpi, program działa niezależnie od procesu weryfikacyjnego Microsoftu. Shim posiada wbudowany system zarządzania certyfikatami, dzięki któremu może je przechowywać pod nazwą MOKs (czyli „klucze właściciela urządzenia” – machine owner keys).

Odzyskiwanie niezależności

Dzięki nowemu rozwiązaniu duży dystrybutorzy, tacy jak Ubuntu, SUSE i Red Hat, odzyskują kontrolę nad sprzętem. Przykładowo, korzystając z certyfikatu Canonicala przechowywanego w Shimie, Ubuntu zatwierdza GRUB-a 2. Oprogramowanie sprzętowe uruchamia Shima, ten z kolei ładuje GRUB-a 2, a GRUB 2 – system operacyjny (Rysunek 2).

Początkowo użytkownik nawet nie zauważa bezpiecznego rozruchu. Na przykład jeśli instalujemy Ubuntu na komputerze, gdzie ten system jest aktywny, cyfrowo podpisane Shim i GRUB 2 umieszczane są na dysku, a następnie instalowane jest, również podpisane, jądro oraz zweryfikowane moduły i sterowniki. Jeśli bezpieczny rozruch nie jest aktywny, w czasie instalacji kopiowane są komponenty bez cyfrowego podpisu.

Wyłączenie bezpiecznego rozruchu

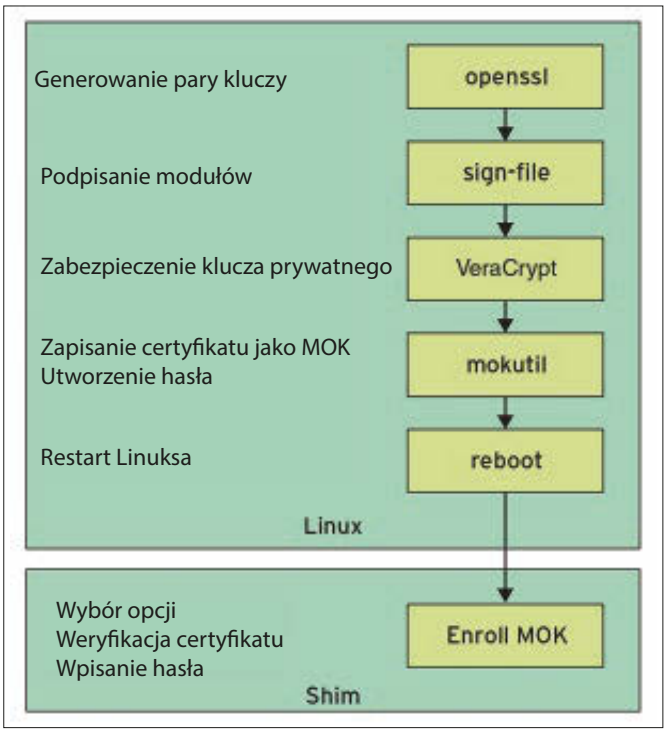
Jeśli spróbujemy zainstalować VirtualBoksa na urządzeniu z bezpiecznym rozruchem i Linuksem, komputer może zaprotestować i nie załadować wymaganego modułu jądra, gdyż nie ma on cyfrowego podpisu. Taka sytuacja zdarza się w pakietach firm trzecich, które dostarczają własnych modułów lub sterowników. Mając fizyczny dostęp do komputera, możemy w dość nieelegancki sposób wyłączyć w Shimie weryfikację podpisów, wydając polecenie:

```
sudo mokutil --disable-validation
```

Narzędzie *mokutil* prosi o zdefiniowanie jednorazowego hasła. Następnie nie wyłącza od razu weryfikacji, ale tak konfiguruje Shima, że ten prosi po następnym restarcie o hasło (dając nam ograniczony czas na wpisanie go) i jeśli jest ono poprawne, pozwala przejść do zmian w konfiguracji.

W oknie, które się pojawi, wybieramy *Change Secure Boot state* (Rysunek 3). Po wpisaniu wcześniej zdefiniowanego

Rysunek 3: Szybko i niebezpiecznie: Shim umożliwia wyłączenie weryfikacji cyfrowych podpisów.



Rysunek 4: Jeśli mamy fizyczny dostęp do komputera, możemy dodawać własne certyfikaty. Ten niezbyt intuicyjny proces przedstawiamy krok po kroku na powyższym rysunku.

jednorazowego hasła, obowiązkowa weryfikacja cyfrowych podpisów zostaje wyłączona. Jednak oznacza to, że tracimy ochronę zapewnianą przez bezpieczny rozruch (cały system możemy też wyłączyć bezpośrednio w ustawieniach UEFI).

Tworzenie własnych certyfikatów

Lepszym sposobem, dodającym elastyczności do naszego systemu bezpiecznego rozruchu, jest przechowywanie własnych certyfikatów w jądrze oraz samodzielne podpisywanie modułów. Problemem jest to, że nowy klucz musi być podpisany z pomocą klucza już istniejącego w bazie certyfikatów, a jedyny klucz, który tam się znajduje, należy do Canonicala. Ponieważ klucz prywatny znany jest tylko w Canonicalu (przynajmniej mamy taką nadzieję), nie możemy wprowadzać modyfikacji. Aby rozwiązać ten problem, generujemy klucze samemu i przechowujemy certyfikaty wykorzystywane w Shimie jako MOK.

Rysunek 4 przedstawia cały proces. Najpierw musimy utworzyć nową parę kluczy. Potem klucz prywatny podpisuje nowe moduły. Następnie zabezpieczamy klucz prywatny i przekazujemy do Shima klucz publiczny jako MOK. Po restarcie pojawia się okienko menedżera certyfikatów Shima. Korzystamy z niego, aby zweryfikować i zaimportować certyfikat. Po następnym restarcie Linux może już korzystać z podpisanych modułów.

Aby utworzyć wymaganą parę kluczy, wydajemy w terminalu następujące, długie polecenie:

```
openssl req -new -x509 -newkey rsa:2048
    -keyout kluczwlasciciela.priv -outform DER
    -out kluczwlasciciela.der -nodes -days 36500
    -subj "/CN=Klucz Właściciela Urządzenia w Mojej Firmie/"
```

KLUCZE, HASŁA I CERTYFIKATY

Nawet eksperci czasami błędnie używają pojęć takich jak klucz, hasło i certyfikat. Klucz to sekwencja bajtów wykorzystywana do szyfrowania danych. Jeśli nasz algorytm korzysta z szyfru blokowego, klucz ma często długość jednego bloku. W przypadku szyfrowania znak po znaku najlepsza długość klucza równa się liczbie znaków, które chcemy zaszyfrować. W przeciwnym wypadku klucz jest używany raz za razem.

Hasło czy klucz. Hasło to ciąg znaków, dzięki któremu powstaje klucz. Na przykład jeśli tworzymy klucz, korzystając z generatora liczb pseudolosowych, podawany na początku ciąg znaków, na podstawie którego następuje generowanie, to nasze hasło.

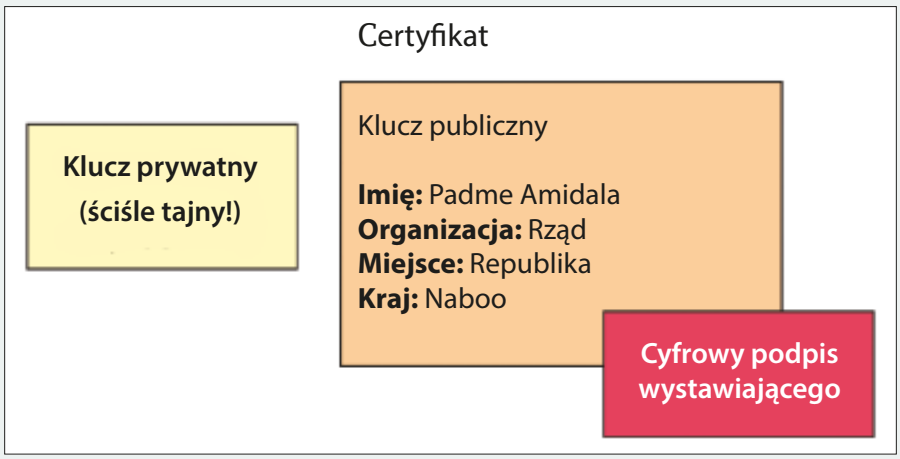
W algorytmie szyfrowania symetrycznego pojedynczym kluczem szyfrujemy i odszyfrowujemy. Taki klucz musi być tajny, znany jedynie wtajemniczonym. Konieczne jest wtedy przesyłanie go do odbiorcy, dlatego powstaje ryzyko odczytania go przez osoby trzecie.

Jednak w algorytmie asymetrycznym posługujemy się parą matematycznie spójnych kluczy. Klucz prywatny jest wykorzystywany do odszyfrowania i podpisywania. Druga strona powinna posiadać klucz publiczny, aby zaszyfrować treść i zweryfikować podpis cyfrowy.

Korzystając z algorytmu asymetrycznego, możemy cyfrowo podpisać daną treść i zaszyfrować ją na potrzeby konkretnych odbiorców. Klucz publiczny, zgodnie z nazwą, nie musi być utrzymywany w tajemnicy i może być bezpiecznie przekazywany.

Klucz z breloczkiem. Dzięki dodaniu do klucza publicznego informacji na temat właściciela w postaci podpisu cyfrowego całość jest odporna na manipulacje. Cyfrowa kombinacja metadanych (nazwa, firma, adres) w połączeniu z kluczem publicznym to certyfikat (Rysunek 5).

Bez względu na to, czy przekazywany jest sam klucz publiczny, czy certyfikat, właściciel powinien go chronić. Powszechne jest szyfrowanie symetrycznie i zabezpieczanie hasłem takiego klucza.



Rysunek 5: Certyfikat składa się z klucza publicznego, metadanych na temat użytkownika klucza prywatnego i cyfrowego podpisu.

W aktualnym katalogu pojawią się klucz prywatny (*kluczwlasciciela.priv*) i certyfikat z kluczem publicznym (certyfikat właściciela *kluczwlasciciela.der*) (więcej w ramce „Klucze, hasła i certyfikaty").

Klucz prywatny powinniśmy przechowywać w sposób bezpieczny, ponieważ każdy, kto może go przeczytać, może też wykorzystać go do cyfrowego podpisu. Często umieszcza się klucze prywatne na pamięciach USB. Dodatkowo zalecamy zaszyfrowanie takich pamięci z pomocą narzędzi typu VeraCrypt [1].

Samopodpisujące się moduły

Niepodpisane moduły powinniśmy podpisać cyfrowo. Służy do tego polecenie *sign-file*, które znajdziemy w źródłach jądra i wywołamy w następujący sposób:

```
sudo /lib/modules/$(uname -r)/build/scripts/sign-file
 sha256 ./kluczwlasciciela.priv ./kluczwlasciciela.der <Nazwa_Modułu>
```

Korzystając ze skryptu z Listingu 1, możemy podpisać wszystkie moduły VirtualBoksa. Formuła *$(modinfo -n vboxdrv)* zwraca ścieżkę do modułu VirtualBoksa (na przykład *vboxdrv.ko*). Ewentualnie możemy też podpisać moduły samemu, podając ścieżkę do każdego z nich, razem z nazwą pliku.

Poniższe proste polecenie przekazuje przykładowy podpis (dostępu do którego nie musimy chronić) do menedżera certyfikatów Shima:

```
sudo mokutil --import kluczwlasciciela.der
```

Listing 1: Podpisywanie modułów VirtualBoksa

```
#!/bin/bash
for drivername in vboxdrv vboxnetflt vboxnetadp vboxpci
do
  sudo /lib/modules/$(uname -r)/build/scripts/sign-file sha256 ./kluczwlasciciela.
priv ./kluczwlasciciela.der $(modinfo -n $drivername)
done
```

Jak zwykle po restarcie musimy podać (w ograniczonym oknie czasowym) jednorazowe hasło, aby przejść do zarządzania certyfikatami.

W celu aktywowania utworzonego dopiero co certyfikatu (Rysunek 6) wybieramy w Shimie pozycję *Enroll MOK*. Z kolei *View Key* sprawdza klucz, a dwa kolejne kroki po wybraniu *Continue* go aktywują.

Po uruchomieniu systemu, polecenie

```
mokutil --list-enrolled
```

zwraca listę zarejestrowanych certyfikatów, łącznie z tymi, które wygenerowaliśmy. Polecenie ładujące moduł jądra VirtualBoksa:

```
sudo modprobe vboxdrv
```

działa już poprawnie. Należy jednak zachować ostrożność: z każdą aktualizacją jądra musimy cyfrowo podpisać nowe moduły z pomocą (miejmy nadzieję) dobrze zabezpieczonego klucza prywatnego.

Rysunek 6: (a) Shim pozwala nam włączyć własne certyfikaty. Wystawiający cyfrowy certyfikat może na pierwszy rzut oka (b) określić, który certyfikat należy do niego (c). (d) Po wybraniu Continue rozpoczyna się proces aktywowania certyfikatu, (e) który potwierdzamy w kolejnym okienku i (f) kończymy, wpisując wcześniej ustalone jednorazowe hasło.

Na koniec przedstawiamy polecenie

```
sudo mokutil --delete kluczwlasciciela.der
```

Usuwa ono certyfikat z menedżera certyfikatów Shima. Odbywa się to przy kolejnym restarcie, jak zawsze jest potwierdzane jednorazowym hasłem, a pozycja menu, którą należy wybrać, to *Delete MOK*.

Lepsze zabezpieczenie, nie oznacza stuprocentowego zabezpieczenia

Instalacja menedżera rozruchu Shim UEFI, cały wysiłek w to włożony, łącznie z podpisywaniem jądra i jego modułów z pewnością polepszają bezpieczeństwo komputera. Dzieciaki ze skryptami znalezionymi w Internecie czy nawet hakerzy mają znacznie trudniejszą przeprawę, chcąc na nim coś zainstalować. Jednak nie do końca mamy powody do świętowania. Kiedy przyjrzymy się bazom certyfikatów, widzimy, że producenci tacy jak Microsoft, Verisign, Symaten i Canonical mają teraz swoje udziały w naszym „bezpiecznym" urządzeniu (Tabela 1).

Po pierwsze, naiwne byłoby myśleć, że wszyscy pracownicy we wszystkich tych firmach na propozycję otrzymania kilku bitcoinów za drobną przysługę odpowiedzą „nie". Po drugie, po otrzymaniu oficjalnego listu z organizacji bezpieczeństwa narodowego wiele koncernów może pójść na ustępstwa w dziedzinie zabezpieczania danych. Koniec końców jest tylko jedna bezpieczna infrastruktura kluczy publicznych: ta, którą sami zarządzamy.

Jeśli przechowujemy nasz klucz w UEFI, pilnujemy istniejących kluczy, kompilujemy i podpisujemy menedżera rozruchu, kompilujemy i podpisujemy jądro, wszystkie moduły oraz sterowniki, możemy podejść do naszego systemu z dużą dozą zaufania. Jeśli mamy większe wymagania co do bezpieczeństwa, kolejnym krokiem będzie zabezpieczenie aplikacji uruchamianych w systemie. ■■■

TABELA 1: Kto co kontroluje

Komponent	Przykładowy podpis	Podpisujący
UEFI (PK, KEK, DB)	Producent, Microsoft (2x)	–
Shim	Dystrybutor (np. Ubuntu)	Microsoft (Verisign/ Symantec)
grubx86.efi	Dystrybutor (np. Ubuntu)	Dystrybutor
Linux Kernel	Dystrybutor (np. Ubuntu)	Dystrybutor

INFO

[1] VeraCrypt (szyfrowanie w czasie rzeczywistym w tle): https://veracrypt.codeplex.com

Zabezpieczanie systemu TPM-em

Zaufana **platforma**

Moduł TPM na naszej płycie głównej może pomóc nam zabezpieczyć system. Matthew Garrett

Bezpieczeństwo każdego systemu operacyjnego zależy od bezpieczeństwa wszystkich warstw znajdujących się poniżej. Jeśli nie mamy zaufania do tego, że procesor we właściwy sposób wykona instrukcje, trudno mówić o jakimkolwiek bezpieczeństwie oprogramowania. Jeśli ktoś majstrował przy menedżerze startu, trudno mieć zaufanie do uruchamianego przez niego jądra. Mechanizm bezpiecznego uruchamiania (*secure boot*) umożliwia weryfikację menedżera startu przed jego uruchomieniem, jeśli jednak w oprogramowaniu sprzętowym znajdują się tylne drzwi, nie uda nam się stwierdzić, czy mechanizm ten zadziałał prawidłowo.

Problem ten wygląda na nierozwiązywalny: możemy zaufać systemowi operacyjnemu, że oprogramowanie sprzętowe jest niezmienione, tylko wtedy, gdy oprogramowanie to faktycznie nie zostało zmienione. Jak możemy zweryfikować stan systemu, nie ufając mu?

Odpowiedzią jest zestaw mechanizmów znanych zbiorczo jako Trusted Computing (TC); ich specyfikacjami zarządza konsorcjum firm o nazwie Trusted Computing Group [1]. W sercu TC znajduje się niewielki komponent sprzętowy o nazwie TPM (skrót od Trusted Platform Module, czyli modułu zaufanej platformy). TPM to układ na płycie głównej dołączony za pomocą prostej szyny. TPM-y nie są ani szybkie, ani potężne – niemal wszystko, co robi TPM, procesor zrobiłby znacznie szybciej. TPM nie widzi też, co dzieje się w pozostałych obszarach systemu – ma jedynie dostęp do informacji, które zostaną mu jawnie przekazane.

Siła TPM-a tkwi w tym, jak wykorzystywane są te informacje. TPM-y mają zestaw rejestrów konfigurujących platformę (Platfom Configuration Registers, PCR – patrz ramka „Zapis w TPM-ie"). Podczas resetowania systemu rejestry te zostają wyzerowane. Podczas uruchamiania komputera system generuje hasze kryptograficzne

ZAPIS W TPM-IE

Kiedy TPM-owi przekazywane są nowe wartości, nie są one zapisywane w PCR-ach bezpośrednio. Zamiast tego PCR ustawiany jest na wartość określoną przez połączenie bieżącej wartości PCR-a i przekazanej mu nowej wartości. Jeśli np. komponentem do pomiaru jest napis *Witaj, świecie!*, natomiast bieżącą wartością PCR-a jest *c72bf7f5a487b1e75819b5b1d1644ded23c10967*, nową wartością będzie:

```
SHA1(c72bf7f5a487b1e75819b5b1d1644ded23c10967 ||
  SHA1("Witaj, świecie!"))
```

Innymi słowy, hasz kryptograficzny SHA1 napisu *Witaj, świecie!* zostaje dodany do bieżącej wartości PCR, a następnie PCR zostaje ustawiony na SHA1 tej połączonej wartości. Oznacza to, że znaczenie ma kolejność pomiarów: pomiar *Idzie wiosna*, a następnie *Przyszła zima* będzie skutkował inną wartością CR niż pomiar *Przyszła zima*, a potem *Idzie wiosna.*

komponentów uruchomieniowych i przekazuje te hasze TPM-owi. Jakakolwiek modyfikacja komponentów uruchomieniowych zmieni wartość tych haszów, zmieniając tym samym wartości zarejestrowane przez TPM. Proces ten nosi nazwę pomiaru, oparte zaś na nim uruchamianie systemu – uruchamiania z pomiarem (Measured Boot).

Pomiary poszczególnych komponentów trafiają do różnych PCR-ów (Tabela 1). Pomiar danego komponentu następuje przed jego wykonaniem, po czym dany komponent przeprowadza pomiar kolejnego; na początku procesor przeprowadza pomiar oprogramowania sprzętowego. Jeśli oprogramowanie sprzętowe było modyfikowane, pojawi się różnica w PCR0. Nawet jeśli później oprogramowanie sprzętowe zafałszuje pomiary późniejszych komponentów (udając, że ich wartości są prawidłowe, podczas gdy w rzeczywistości zostały zmodyfikowane), wartość PCR0 będzie nadal różna. Patrząc na wartości PCR i porównując je z wartościami oczekiwanymi, możemy ustalić, czy ktoś czegoś nie majstrował przy którymś z etapów procesu uruchomieniowego.

Jak jednak ocenić te wartości? Jeśli system operacyjny został zmodyfikowany, nie możemy mieć pewności, że przekaże nam wartości, które faktycznie zostały zachowane w TPM-ie. Potrzebujemy więc uzyskać pomoc ze strony samego TPM-a. W ten sposób dochodzimy do kolejnej funkcji udostępnianej przez TPM-y: generowanie i przechowywanie kluczy szyfrujących.

Podczas inicjalizacji TPM generuje klucz główny (Storage Root Key, SRK). Klucz ten nigdy nie opuszcza TPM-a, system operacyjny nie ma zaś do niego dostępu. Kiedy aplikacja prosi TPM-a o wygenerowanie klucza, TPM robi to i przekazuje aplikacji połówki publiczne i prywatne. Zanim jednak przekaże prywatną połówkę, TPM zaszyfruje ją za pomocą SRK. Klucz prywatny może zostać użyty wyłącznie po przekazaniu go z powrotem do TPM-a, który odszyfruje go i zachowa w pamięci wewnętrznej TPM-a. Wszelkie dane zaszyfrowane publiczną połówką można odszyfrować wyłącznie TPM-em, który potrafi odczytać powiązany klucz prywatny: jeśli mamy parę kluczy TPM, a ktoś ukradnie nam dysk, złodziej nie będzie mógł użyć tych kluczy, ponieważ może je odszyfrować wyłącznie TPM, który je wygenerował.

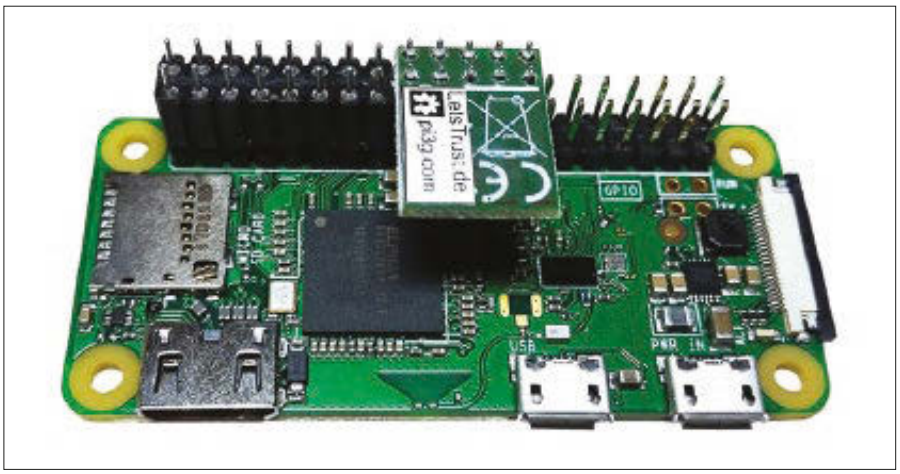
Kiedy aplikacja poprosi o dane zaszyfrowane tym kluczem, może podać zestaw wartości PCR, które mają zostać powiązane z szyfrowanymi danymi. Jeśli podczas deszyfracji owe wartości PCR nie zgadzają się, TPM odmówi odszyfrowania. Jeśli używamy TPM-a do szyfrowania klucza, za pomocą którego zaszyfrowaliśmy dysk twardy, nasz komputer nie uruchomi się, jeśli którykolwiek z komponentów procesu uruchamiania został zmodyfikowany [2]. Jeśli zamiast BIOS-u używamy UEFI, powinniśmy zamiast TrustedGRUB2 użyć gałęzi Gruba o nazwie *verifier_tpm_module*.

W takiej konfiguracji system, przy którym ktoś majstrował, nie zostanie uruchomiony. Nie znaczy to jednak, że włamywacz, który ma fizyczny dostęp do sprzętu, nie będzie mógł uzyskać haseł. Może po prostu usunąć dysk twardy i zastąpić go takim, który uruchamia się normalnie, nawet jeśli proces uruchamiania został zmodyfikowany. Kiedy będziemy próbowali się zalogować, system zachowa po prostu nasze hasło i prześle je atakującemu. Można tego uniknąć, stosując Tpmtotp [4], który wykorzystuje protokół czasowych haseł jednorazowych (TOTP) używany przez różne witryny do uwierzytelniania dwuetapowego (2FA). Sekretna wartość zostaje zaszyfrowana na TPM-ie i może zostać odszyfrowana jedynie wtedy, gdy zgadzają się wartości CR. Sekretna wartość jest następnie wyświetlana w formie kodu QR, który można wprowadzić w zwykłe aplikacji 2FA. Podczas uruchamiania system odszyfrowuje sekretną wartość i używa jej w połączeniu z bieżącym czasem do wyświetlenia sześciocyfrowego kod. Następnie użytkownik porównuje ten kod z liczbą wyświetlaną na telefonie. Jeśli obie wartości się zgadzają, użytkownik wie, że systemowi można zaufać, i wprowadza swoje hasło (patrz ramka „Opcja z RAM-em").

Rzecz jasna, nawet potrzebne i zalecane aktualizacje komponentów uruchomieniowych spowodują zmianę wartości PCR. Przykładowo aktualizacja GRUB-a zmieni wartość PCR4. Tak więc system staje się bardzo wrażliwy: zwykle poprawki związane z bezpieczeństwem mogą spowodować, że system przestanie się uruchamiać. Jedną z metod ominięcia tego problemu jest używanie TPM-a razem z mechanizmem bezpiecznego uruchamiania (SB). W tym trybie używany jest wyłącznie PCR7. Oprogramowanie sprzętowe rejestruje w PCR7 fakt, że włączone jest SB, a następnie przeprowadza pomiar klucza SB używanego podczas uruchamiania systemu. Modyfikacja menedżera startu lub jądra wymagałaby teraz od atakującego wyłączenia SB lub dodania nowych kluczy, co zmieniłoby wartość PCR7. Normalne aktualizacje zostaną podpisane tym samym kluczem, zatem wartość ta pozostanie niezmieniona.

TABELA 1: PCR-Y

PCR	Pomiary
PCR0	Pomiary oprogramowania sprzętowego
PCR1	Pomiary konfiguracji oprogramowania sprzętowego
PCR2	Pomiary oprogramowania sprzętowego karty
PCR3	Pomiary konfiguracji oprogramowania sprzętowego karty
PCR4	Pomiary tablicy partycji i menedżera uruchamiania
PCR5	Pomiary konfiguracji menedżera uruchamiania
PCR6	Pomiary zdarzeń uśpienia i wznowienia pracy systemu
PCR7	Pomiary konfiguracji bezpiecznego uruchamiania



Rysunek 1: LetsTrust udostępnił moduł TPM dla Raspberry Pi.

OPCJA Z RAM-EM

Alternatywne rozwiązanie polega na wykorzystaniu niewielkiej ilości nieulotnego RAM-u układu TPM. Możemy skonfigurować system w taki sposób, by nieulotny RAM dało się odczytać tylko wtedy, gdy wartości PCR pasują do siebie: jeśli nie, próba odczytu się nie powiedzie. W ten sposób sekretną wartość można zachować bezpośrednio, nie martwiąc się o szyfrowanie. Minusem tej techniki jest ograniczona ilość dostępnej pamięci *nvram*.

Dzięki temu użytkownicy mogą mieć większą pewność, że korzystanie z TPM-ów jest bezpieczne i nie muszą się martwić, że system nagle przestanie działać.

Klucze szyfrujące TPM mogą być wykorzystywane nie tylko podczas uruchamiania systemu. Można wygenerować klucze SSH powiązane z TPM-em [5], co zwiększa ochronę: nawet jeśli komuś uda się ukraść nasz klucz prywatny SSH, włamywacz nie zaloguje się do systemu, jeśli nie uzyska dostępu do TPM-a, który mógłby ten klucz odszyfrować.

TPM-y a DRM

Kiedy ponad dziesięć lat temu po raz pierwszy pojawiło się Trusted Computing, wyrażono ogromne obawy, że mechanizm ten zostanie użyty w celu ograniczania dostępu do witryn i usług, jeżeli użytkownik nie korzysta z odpowiedniej wersji Windows. Największym wyzwaniem była zdalna atestacja. Każdy TPM ma klucz zwany kluczem zatwierdzającym (endorsement key, EK).

Zdalna witryna może poprosić system operacyjny o przeprowadzenie zdalnej atestacji. System operacyjny poprosi wtedy TPM-a o podanie bieżących wartości PCR i zaszyfruje je za pomocą EK. Zdalna witryna odszyfruje je, sprawdzi wartości PCR i w zależności od wyniku weryfikacji postanowi, czy udzielić dostępu.

W praktyce nie należy się tym przejmować z dwóch powodów. Po pierwsze, aby ów schemat zadziałał, zdalna witryna musi wiedzieć, że EK faktycznie jest w TPM-ie. Nowoczesne TPM-y zawierają certyfikat EK, który zapewnia łańcuch zaufania od TPM-a do producenta TPM-a, co oznacza, że zdalna witryna może sprawdzić, czy wartości PCR pochodzą z TPM-a, nie wie jednak, z którego, chyba że użytkownik w jakiś sposób zarejestrował uprzednio tę relację. Wiąże się z tym kolejny problem: nikt nie broni użytkownikowi dodać do systemu drugiego TPM-a, programując CR-y „dobrymi" wartościami, a następnie przeprowadzając zdalną atestację drugim TPM-em.

Ze względu na te problemy (i inne kwestie związane z prywatnością) zdalna atestacja nigdy nie była używana poza bardzo ograniczonymi scenariuszami i jest bardzo mało prawdopodobne, by sytuacja ta uległa zmianie. Włączenie TPM-a nie spowoduje więc zagrożenia dla naszej wolności.

Korzystamy z TPM-a

Korzystanie z TPM-a wymaga kilku rzeczy. Po pierwsze, TPM musi być włączony, po drugie, musi mieć odpowiedni menedżer uruchomieniowy, a po trzecie, musi mieć niezbędne narzędzia w przestrzeni użytkownika. TPM-a włączamy w menu naszego oprogramowania sprzętowego, dokładne instrukcje znajdziemy w dokumentacji do płyty głównej. Jeśli chodzi o menedżery startu, będziemy potrzebowali zmodyfikowanej wersji GRUB2 [6]. Oprócz tego powinniśmy zainstalować TrouSerS (demon i biblioteka TPM) oraz pakiet *tpm-tools*. Kontrolę nad swoim TPM-em przejmujemy poleceniem:

```
sudo tpm_takeownership -z
```

Jeśli otrzymamy komunikat o błędzie „The TPM target command has been disabled" (docelowe polecenie TPM zostało wyłączone, oznacza to, że TPM został już włączony. Jeśli to nie my go włączyliśmy, możemy zresetować TPM-a poleceniem:

```
echo 14 >/sys/class/tpm/tpm0/ppi/request
```

Wykonujemy je jako administrator i restartujemy system. Oprogramowanie sprzętowe zapyta nas, czy chcemy wyczyścić TPM-a. Wykonujemy instrukcje na ekranie i ponownie próbujemy przejąć kontrolę nad TPM-em.

TPM 1.2 i TPM 2

Od niedawna w komputerach często umieszczane są TPM-y zgodne ze specyfikacją w wersji 2. Została ona znacznie ulepszona w stosunku do wersji 1; zawiera m.in. bardziej nowoczesne i bezpieczne algorytmy haszujące. Niestety, obsługa urządzeń TPM2 w Linuksie nadal raczkuje, zaś większość oprogramowania opisanego w tym artykule jest kompatybilna wyłącznie ze starszą wersją, czyli TPM 1.2.

W niektórych systemach TPM 1.2 znajduje się na płycie głównej, natomiast TPM zaimplementowany jest w formie emulowanego TPM-a działającego na zintegrowanym z procesorem mechanizmie Management Engine. W tym przypadku powinniśmy sprawdzić obecność w ustawieniach opcji „Intel PTT", „Intel Platform Trust Technology" lub „Firmware TPM" i ją wyłączyć.

Niektórzy producenci zaimplementowali inne rozwiązanie: udostępniają na płycie sprzętowy TPM, na który można wgrać wersję 1.2 lub 2.0; zrobił tak np. Dell z XPS 13. Jeśli w naszym systemie zainstalowany jest TPM 2.0, warto zapytać producenta, czy da się skorzystać z wersji 1.2.

Wnioski

Dobre wykorzystanie TPM-a pod Linuksem nadal wymaga znacznych nakładów ręcznej pracy, jest ona jednak tego warta. Można oczekiwać, że w niedalekiej przyszłości omawiane funkcje zostaną zintegrowane z dystrybucjami, dzięki czemu nie tylko eksperci, ale także zwykli użytkownicy będą mogli korzystać z zaawansowanych zabezpieczeń systemu. Jeśli wszystko dobrze pójdzie, każdy będzie mógł korzystać z TPM-ów. ■■■

INFO

[1] Trusted Computing Group: https://trustedcomputinggroup.org/
[2] Jak używać TPM-a do szyfrowania: https://github.com/fox-it/linux-luks-tpm-boot
[3] GRUB: https://github.com/mjg59/grub
[4] Tpmtotp: https://github.com/mjg59/tpmtotp
[5] Konfiguracja SSH z TPM-em: https://www.arm-blog.com/using-the-tpm-module-for-ssh-key-signing/
[6] GRUB2: https://github.com/mjg59/grub/tree/verifier_tpm_module

Automatyzacja kopii zapasowych w wierszu polecenia

Automatyczna kopia zapasowa

Wykonywanie kopii zapasowych nie należy do ulubionych zajęć użytkowników, a nawet niektórych administratorów – bywa traktowane jak sprzątanie czy zmywanie. Z tego powodu zainteresowaliśmy się programami do wykonywania kopii zapasowej działającymi w wierszu polecenia. Erik Bärwaldt

Na Linuksa dostępnych jest wiele narzędzi do tworzenia kopii zapasowych. Administratorom pracującym z SSH spodoba się fakt, że programami działającymi w wierszu polecenia można wykonywać kopie zapasowe serwerów o dowolnym rozmiarze i budowie. Znaczące są jednak różnice w funkcjonalności tych programów (streszczenie w Tabeli 1). Nie każdy program nadaje się do każdego zastosowania, więc przyjrzymy się dokładniej, które narzędzie sprawdzi się w jakim środowisku.

Tabela 1: Konsolowe narzędzia do tworzenia kopii zapasowych

	Attic	bup	Duplicity	rdiff-backup	rsnapshot
Kopia lokalna	Tak	Tak	Tak	Tak	Tak
Kopia przez SSH	Tak	Tak	Tak	Tak	Tak
Weryfikacja	Tak	Tak	Tak	Tak	Tak (plik dziennika)
Szyfrowanie	Tak	Tak	Tak	Nie	Nie
Współpraca z chmurą	Nie	Nie	Tak (Amazon, Rackspace)	Nie	Nie
Włącz/wyłącz katalog	Tak	Tak	Tak	Tak	Tak
Kontrola czasowa	Tak*	Tak*	Tak*	Tak*	Tak*
Dostępność nakładek graficznych	Nie	Tak	Tak	Nie	Tak
Kopia przyrostowa	Tak	Tak	Tak	Tak	Nie
Kopia różnicowa	Nie	Nie	Nie	Nie	Nie
Pełna kopia ręczna	Tak	Tak	Tak	Tak	Tak
Montowanie w przestrzeni użytkownika	Tak	Tak	Nie	Tak	Nie

*Planowanie operacji kopii demonem cron

Serwer a użytkownik domowy

Użytkownicy domowi często przechowują na komputerach dane w ilościach nieodbiegających od zajętości serwerów w małych firmach. Zbiory filmów w wysokiej rozdzielczości, bezstratne pliki audio czy foldery ze zdjęciami potrafią zużyć mnóstwo pamięci. Do takiego zbioru nowe dane dochodzą często, a w starych rzadko zachodzą zmiany.

W systemach serwerowych znajdziemy natomiast wiele małych plików (korespondencja, tabele, prezentacje, bazy danych). Te zbiory ulegają ciągłym zmianom, takim jak nowe rekordy w bazie czy właśnie

dodane dokumenty. W strategiach kopii zapasowych trzeba więc uwzględniać charakter zastanych zbiorów, aby w razie utraty danych można było zagwarantować ich szybkie odtworzenie.

Kopia różnicowa a przyrostowa

Rozróżnia się trzy podejścia do kopii zapasowej: kopia pełna, różnicowa i przyrostowa. Każda z tych strategii zaczyna się od kopii pełnej, czyli skopiowania wszystkich zastanych danych. Kopie następujące później są różnicowe lub przyrostowe. W kopii różnicowej zapisują się wszystkie zmiany względem kopii pełnej, kopia przyrostowa zaś zachowuje jedynie zmiany powstałe od ostatniej operacji kopii zapasowej.

Proces różnicowy wymaga więcej miejsca na poszczególne kopie, ale w razie nieszczęśliwego wypadku do odzyskania całej bazy wystarczy kopia pełna i najnowsza różnicowa. Kopia przyrostowa korzysta z mniejszej ilości miejsca, ale przy odzyskiwaniu należy użyć pliku kopii pełnej i wszystkich pozostałych plików we właściwej kolejności. Jeśli w sekwencji zabraknie choćby najmniejszego pliku, odtworzenie bazy danych nie będzie już możliwe.

Przed zdecydowaniem się na konkretny konsolowy program polecam uważne zanalizowanie kolekcji danych i jej rozrostu, aby przypadkiem nie wybrać narzędzia nienadającego się do konkretnego środowiska.

Wykonywanie kopii różnicowych będzie sensowne w przypadku baz danych ze względnie niedużą liczbą dużych plików i raczej regularnymi modyfikacjami, w typowych środowiskach biurowych zaś lepiej sprawdzą się kopie przyrostowe. Niezależnie od wybranej strategii powinno się zawsze wykonywać co najmniej jedną pełną kopię tygodniowo.

Użytkownicy desktopów chcący skopiować własne zbiory bez przywilejów administratora nie mają niestety dużego wyboru narzędzi konsolowych, a praca z nimi wymaga przecież znajomości składni poleceń. Z punktu widzenia użytkownika najważniejsze jest, by wykonywanie kopii przebiegało możliwie sprawnie i bezawaryjnie. Użytkownicy końcowi zdecydują się na dany program i będą w nim faktycznie robić kopie, tylko gdy będzie to łatwa i szybka operacja.

Idealny program tego typu można by wykorzystać zarówno w środowisku mieszanym zarówno z serwerem kopii zapasowych, jak i do dodatkowych kopii na pulpitach użytkowników. Używanie jednego programu do wszystkiego oszczędza nam żmudnej nauki składni dwóch narzędzi i związanego z tym ryzyka błędów.

Attic

Program Attic (ang. *strych*, *poddasze*) napisany jest w Pythonie i znajduje się w repozytoriach niektórych dystrybucji, takich jak Mageia, openSUSE, ROSA czy Slackware. Można go wygodnie zainstalować z poziomu odpowiedniego menedżera pakietów lub pobrać kod źródłowy z witryny projektu, na której znajdziemy także szczegółową dokumentację [1].

Aby Attic zadziałał, w systemie potrzebny jest Python w wersji 3.2 lub wyższej oraz OpenSSL nowszy niż wersja 1.0.0. Program umożliwia także montowanie zbioru kopii w przestrzeni użytkownika, ta funkcja zaś wymaga pakietu *llfuse* z bogatej kolekcji rozszerzeń Pythona.

Pierwszym krokiem po zakończeniu instalacji jest inicjalizacja nowego repozytorium kopii zapasowej. Robi się to poleceniem:

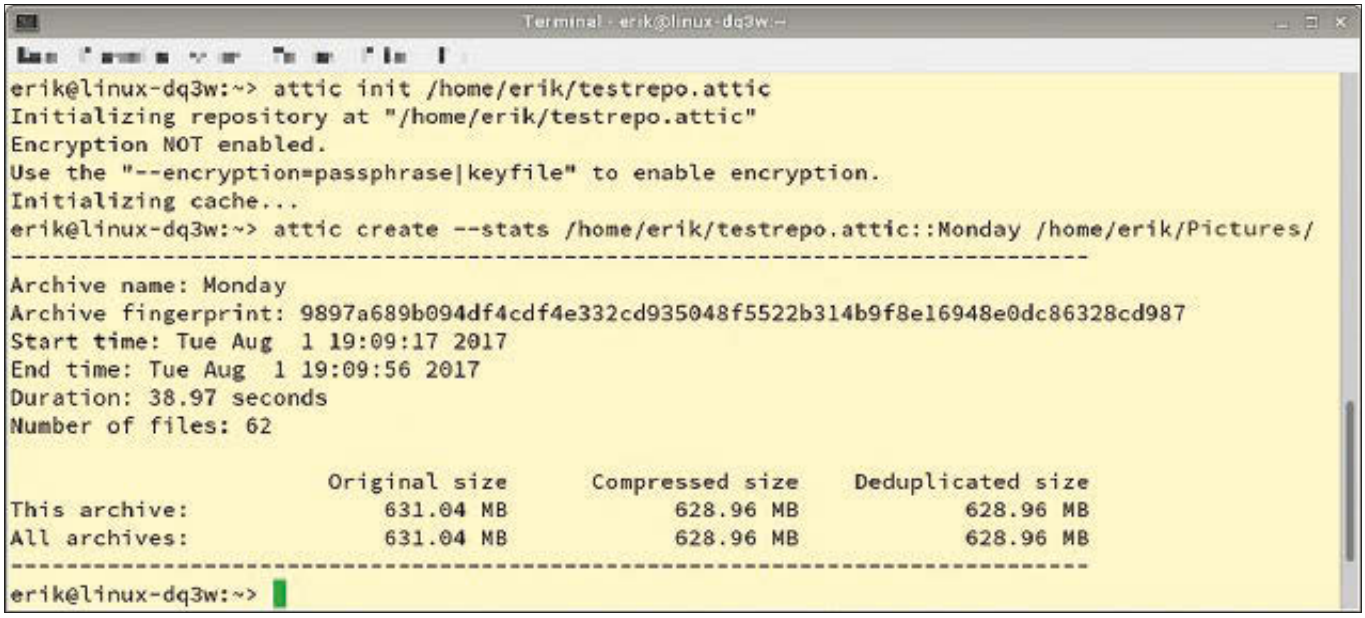
```
attic init /<ścieżka-repozytorium>/
 <nazwa-repozytorium>.attic
```

Teraz w repozytorium można zachować dane z kilku lokalizacji, które znajdą się w specjalnie stworzonym archiwum. Domyślnie Attic nie udostępnia opcji szyfrowania danych. Nazwy naszych archiwów mogą być zupełnie dowolne. Kopię zapasową lokalizacji wykonujemy poleceniem:

```
attic create /<ścieżka-repozytorium>/
 <nazwa-repozytorium>.attic::
 <nazwa archiwum>/
 <katalog źródłowy 1><[...]>
/<katalog źródłowy n>
```

Jeśli dane muszą być zaszyfrowane, dodajemy polecenie z parametrami:

```
--encryption=<hasło | klucz>
```



Rysunek 1: Attic przejrzyście informuje o ostatniej kopii zapasowej.

Przy regularnych kopiach tych samych katalogów polecam nazywać archiwa według dni tygodnia, a w momencie przywracania kopii szybko ustalimy ich poprawną sekwencję. Przy dużych woluminach tworzenie pierwszej kopii do danego repozytorium może trwać długo, ale kolejne zgrają się dużo szybciej, ponieważ Attic archiwizuje przyrostowo (tzn. do kopii trafiają tylko dane nowe lub zmodyfikowane).

Jeśli chcemy nadzorować tworzenie kopii, najważniejsze informacje o zapełnianym właśnie archiwum wyświetlimy parametrem *--stats*. Attic wypunktuje kopiowane przez siebie lokalizacje i czas ich przetwarzania, a także liczbę kopiowanych plików i objętość danych. Tę ostatnią podaje zarówno dla oryginałów, jak i kopii, więc można prześledzić wydajność kompresji (Rysunek 1).

W przeciwieństwie do wielu innych narzędzi do kopii zapasowych, w Attic można wygodnie wyświetlić listę zawartości archiwum. W tym celu wpisujemy polecenie:

```
attic list -v /<ścieżka-repozytorium>/
 <nazwa-repozytorium>.attic::
 <nazwa-archiwum>
```

Program wymieni wtedy całą zawartość kopii, w tym rozmiar plików i uprawnienia do nich, z automatycznie uwzględnionymi podfolderami. Widać także ścieżki bezwzględne.

Weryfikacja archiwum

W Atticu podobnie prosto sprawdza się spójność danych. Polecenie

```
attic check /<ścieżka-repozytorium>/
 <nazwa-repozytorium>.attic
```

sprawdza repozytorium oraz wszystkie jego archiwa. O stanie każdego elementu informuje krótki komunikat (Rysunek 2). Jeśli pojawią się niespójności, dane można naprawić, powtarzając polecenie z dodatkowym parametrem *--repair*.

Przywracanie

Archiwum przywracamy, wybierając opcję *extract*. Polecenie

```
attic extract /<ścieżka-repozytorium>/
 <nazwa-repozytorium>.attic:
 <nazwa-archiwum>
```

uruchamia przywracanie całej zawartości archiwum do oryginalnej lokalizacji, a program wylicza poszczególne przenoszone pliki. Docelową ścieżkę można zmienić na inną. Da się także wyłączyć fragmenty archiwum z operacji przywracania.

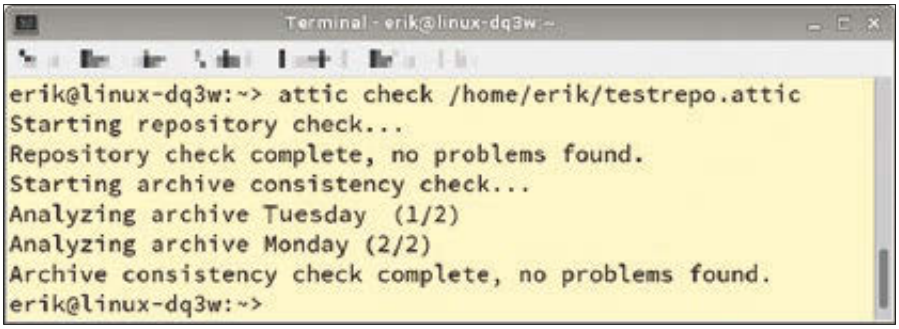
Attic może też oczywiście przechowywać kopie zapasowe na zdalnym serwerze i stamtąd pobierać je do odzyskania. Składnia jest taka sama, jak przy kopii lokalnej, a z serwerem komunikujemy się jak poniżej:

```
<użytkownik>@<nazwa-serwera>:
 <nazwa-repozytorium>.attic
```

Przy tworzeniu repozytoriów na systemie serwerowym zaleca się włączenie szyfrowania. Attic przy szyfrowaniu korzysta z 256-bitowego AES, przy weryfikacji zaś – z HMAC-SHA256, a całość danych szyfruje przed zapisaniem do archiwum.

Automatyzacja

Program potrafi przeprowadzać operacje tworzenia kopii sterowane i zautomatyzowane z poziomu Crona. Aby na dłuższą metę liczba repozytoriów nie wymknęła się spod kontroli, parametrem *prune* w Atticu ograniczamy czas przechowywania starszych repozytoriów, ale w postaci maksymalnej liczby archiwów w danym repozytorium. Można określić, czy chodzi nam o archiwa tworzone co godzinę, dobę, tydzień lub miesiąc. Wcześniej należy jednak wygenerować jakieś archiwa z parametrem *date*.



Rysunek 2: Od razu widać, czy wszystkie dane zostały prawidłowo zapisane.

bup

Program bup (skrót od *backup*), który powstał siedem lat temu i wciąż jest rozwijany, ugruntował swoją pozycję w niedużych infrastrukturach informatycznych ze względu na szybkość i skuteczność [2]. Program jest natychmiast gotowy do instalacji z repozytoriów wielu głównych dystrybucji Linuksa.

Bup różni się od innych narzędzi do kopii zapasowych już samym formatem plików – zamiast tradycyjnych archiwów zip czy tar tworzy spakowane pliki Gita (*packfiles*). Program jest bardzo elastyczny w zastosowaniach – jego archiwa można montować w przestrzeni użytkownika jako systemy plików. Nie ma też żadnych problemów z kopiowaniem całych obrazów ISO, zawierających często setki gigabajtów danych. Aby móc jednak użyć formatu Gita, trzeba najpierw przygotować jego repozytorium.

Aby zobaczyć przegląd licznych parametrów Bupa, wywołujemy go w terminalu bez żadnych opcji. Program wymieni wtedy najważniejsze parametry w postaci krótkiej. Szczegółową dokumentację zawiera natomiast witryna projektu [3]. Przy wykonywaniu kopii na pulpitach przydadzą się nakładki graficzne oparte na Gtk3 i Qt, ułatwiające pracę mało doświadczonym użytkownikom – również znajdziemy je na stronie internetowej Bupa.

Korzystanie

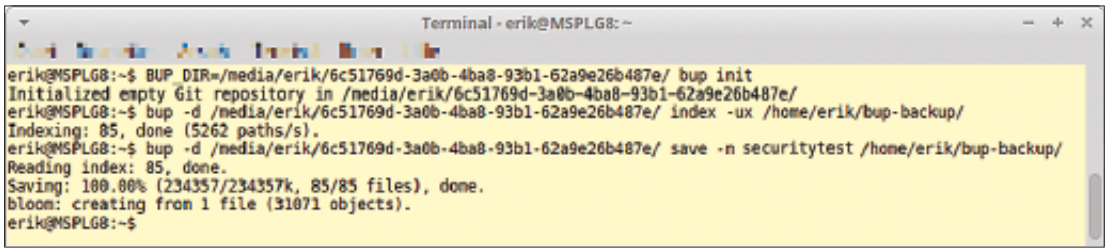
Aby stworzyć kopię, inicjujemy najpierw katalog na nasze zbiory:

```
BUP_DIR=/<katalog-zbioru>bup init
```

Następnie indeksujemy katalog, którego kopie zapasowe chcemy tworzyć:

```
bup index -ux /<ścieżka>
```

Jeśli do danego katalogu nie ma określonej zmiennej ze ścieżką na kopie, to do jego ścieżki każdorazowo trzeba dołączać ścieżkę na kopie pod parametrem *-d*. W przeciwnym razie bup zapisuje zbiór z kopią do ukrytego podkatalogu



Rysunek 3: Bup wykonuje kopię danych w kilku krokach.

.bup w naszym katalogu domowym. Po zaindeksowaniu przystępujemy do tworzenia kopii:

```
bup save -n <nazwa-katalogu-zbioru> /
  <ścieżka-katalogu-do-skopiowania>
```

Jeśli brakuje zmiennych określających ścieżki, to położenie kopii również trzeba określić pod parametrem *-d*. Nazwę zbioru z kopią definiuje się opcjonalnie, ale przyda się do namierzenia właściwego zbioru, jeśli na jednym nośniku docelowym znajduje się kilka kopii zapasowych.

Podczas pierwszej operacji wykonywania kopii bup tworzy kopię pełną – w zależności od rozmiarów woluminu może to trochę potrwać. W kolejnych, dużo szybszych operacjach powstają już tylko kopie przyrostowe.

Trzeba pamiętać, że jeśli w oryginalnej zawartości zaszły zmiany, to przed kolejnym uruchomieniem kopiowania należy ponownie ją zindeksować, aby program mógł wykryć zmiany.

Jeśli chcemy przechowywać kopię na zdalnym serwerze, w najprostszym przypadku wystarczy sekwencja:

```
bup save -r <użytkownik@ip-serwera>
  <katalog-kopii> -n <nazwa-zbioru> /
  <katalog-do-skopiowania>
```

W tym przypadku za pierwszym razem wykonuje się kopia pełna, a kolejne są przyrostowe (Rysunek 3).

Za pomocą bupa można także skopiować zawartość zdalnej maszyny na maszynę lokalną. W tym celu wpisujemy polecenie *bup on*, a po nim adres serwera oraz katalog docelowy na maszynie lokalnej. Aby operacja się udała, wymagane jest oczywiście wcześniejsze zindeksowanie zawartości maszyny.

Aby zweryfikować istniejące zbiory kopii zapasowych, listujemy poszczególne pliki poleceniem *bup ls*, a dostaniemy przejrzysty podgląd w kolumnach (Rysunek 4).

Na wstecznym

Przywracanie z kopii jest równie łatwe dla użytkowników. Po podaniu ścieżki do zbioru z kopiami, zamiast parametru *save* używamy *restore*, a po nim wpisujemy nazwę zbioru i docelową ścieżkę. Istnieje również możliwość przywrócenia poszczególnych podkatalogów ze zbioru.

Dodatki

Oprócz podstawowych funkcji bup oferuje jeszcze kilka możliwości sprawdzania spójności zbiorów kopii. Jednym z miłych dodatków jest opcja montowania kopii zapasowej w przestrzeni użytkownika, podobnie jak robi się to ze zwykłym napędem. Aby funkcja działała, w systemie musi być ścieżka montowania oraz zainstalowany pakiet *python-fuse*. Zbiór włączamy do systemu poleceniem:

```
bup -d <ścieżka-do-zbioru> fuse
  <ścieżka-docelowa>
```

I już można korzystać z katalogu docelowego i jego zawartości jak z każdego innego zamontowanego napędu.

Duplicity

Duplicity [4], dostępny jako pakiet binarny na niemal wszystkie popularne dystrybucje Linuksa, jest bardzo elastyczny w kwestii lokalizacji kopii. Można je zapisywać lokalnie, ale także na FTP czy serwerach SSH w intranecie. Duplicity obsłuży ponadto udziały Windows i serwery WebDAV.

Jeśli centralna lokalizacja na wszystkie kopie zapasowe nie jest dostępna, Duplicity potrafi przechowywać kopie w chmurze. Poradzi sobie z chmurami S3 i Rackspace Amazonu. Jednym z niekwestionowanych atutów Duplicity jest szyfrowanie: wszystkie pliki można szyfrować GnuPG (GPG) oraz bezpiecznie przechować w chmurze bez obaw o podglądanie.

Program prezentuje bardzo profesjonalne podejście do tworzenia kopii zapasowych. Za pierwszym podejściem wykonuje pełną kopię do archiwum tar, a następnie wykonuje kopie przyrostowe. Oszczędza to miejsce na nośniku, ale w przypadku większych baz – również czas. Nawet w mało bezpiecznych środowiskach spójność danych gwarantowana jest podpisem elektronicznym.

Przy wszystkich swoich zaletach Duplicity wymaga od użytkownika długiego przeszkolenia ze względu na skomplikowane parametry [5]. Zalecaną alternatywą mogą być bardziej intuicyjne nakładki graficzne takie jak Déjà Dup [6], szczególnie w małych środowiskach lub na desktopach. W porównaniu z konsolowym Duplicity trzeba się jednak liczyć ze znacznym ograniczeniem funkcjonalności. Ponadto ten program nie obsługuje kopii różnicowych.

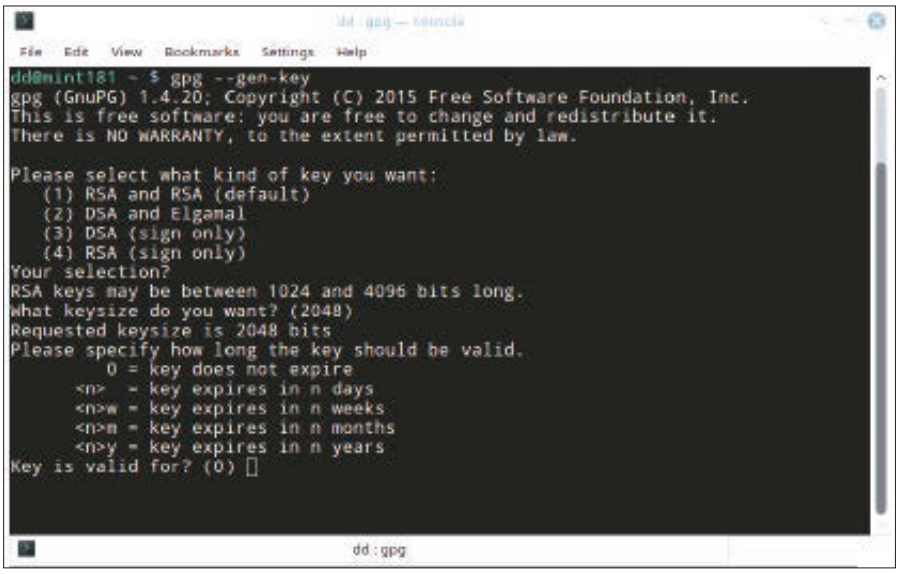
Duplicity nadaje się do zabezpieczania poszczególnych folderów, ale już nie do tworzenia obrazów systemu. (Do tego celu lepiej nadają się programy typu Clonezilla [7].) Duplicity trzyma skopiowane przez siebie dane w postaci woluminów. Jeśli robimy kopię lokalnie, trzeba określić bezwzględną ścieżkę do odpowiedniego docelowego folderu.

Pod kluczem

Aby Duplicity wykonał swój popisowy numer, kopię szyfrowaną, należy najpierw wygenerować klucz GPG lub już taki posiadać. Klucz generujemy w kilku krokach poleceniem *gpg --gen-key* i można samodzielnie określić jego siłę (Rysunek 5).

Bezpieczeństwo naszych danych zwiększa hasło podawane przy

Rysunek 4: Prosty podgląd danych zabezpieczonych w kopii ułatwia administrom sprawdzenie jej kompletności.



Rysunek 5: Klucz GPG generuje się w kilka minut.

generowaniu klucza: jeśli zgubimy klucz, ktoś obcy nie będzie mógł odszyfrować archiwum, ponieważ przy przywracaniu z kopii wymagane jest hasło. Użytkownicy, którzy wyjątkowo drżą o bezpieczeństwo swoich danych, mogą jeszcze podpisać archiwa, aby później można było sprawdzić spójność woluminów z kopią.

W najprostszym przypadku kopię lokalną wykonamy poleceniem:

```
duplicity /<katalog-źródłowy>file:
 //<katalog-docelowy>
```

Po dwukrotnym monicie o hasło program tworzy kopię w katalogu docelowym.

Aby zapisać dane na serwerze FTP, wywołujemy program poleceniem:

```
duplicity /<katalog-źródłowy>
 ftp://<użytkownik>@<nazwa-hosta>/
 <katalog-docelowy>
```

Tutaj również przed zapisaniem zostaniemy poproszeni o hasło. Monit można pominąć, jeśli do polecenia zgrania kopii dodamy ciąg *FTP_PASSWORD=<hasło>*. W obu przypadkach Duplicity automatycznie sprawdza, czy w ścieżce docelowej istnieje już pasująca pełna kopia. Jeśli jej nie ma, za pierwszym razem ją tworzy, a następnie dołącza kopie przyrostowe (Rysunek 6).

Manual Trigger

Z czasem na docelowym nośniku ląduje coraz więcej kopii przyrostowych, szczególnie jeśli operacje kopiowania są zautomatyzowane. Ponieważ trzeba pilnować sekwencji kopii przyrostowych od najnowszej aż do pełnej kopii, dobrze jest regularnie tworzyć aktualniejszą kopię pełną. Zmniejszy to liczbę archiwów z kopiami przyrostowymi. Duplicity zadziała w trybie pełnej kopii ręcznej, jeśli uruchomimy go z parametrem *full*. Podanie parametru *incremental* spowoduje natomiast ręczne wykonanie kopii przyrostowej.

Jeśli jakieś katalogi mają nie być objęte kopią, można określić je parametrem *--exclude*, rozdzielając spacją. Wykluczenie pojedynczych folderów to przydatna opcja, zwłaszcza dla podfolderów zawierających pliki tymczasowe, cache albo pliki dziennika.

Do kopii zapasowej katalogu roota trzeba podejść bardzo ostrożnie i koniecznie wykluczyć katalog */proc* – w przeciwnym wypadku duplicity się zawiesi. Dodatkowe foldery dodamy natomiast parametrem *--include*.

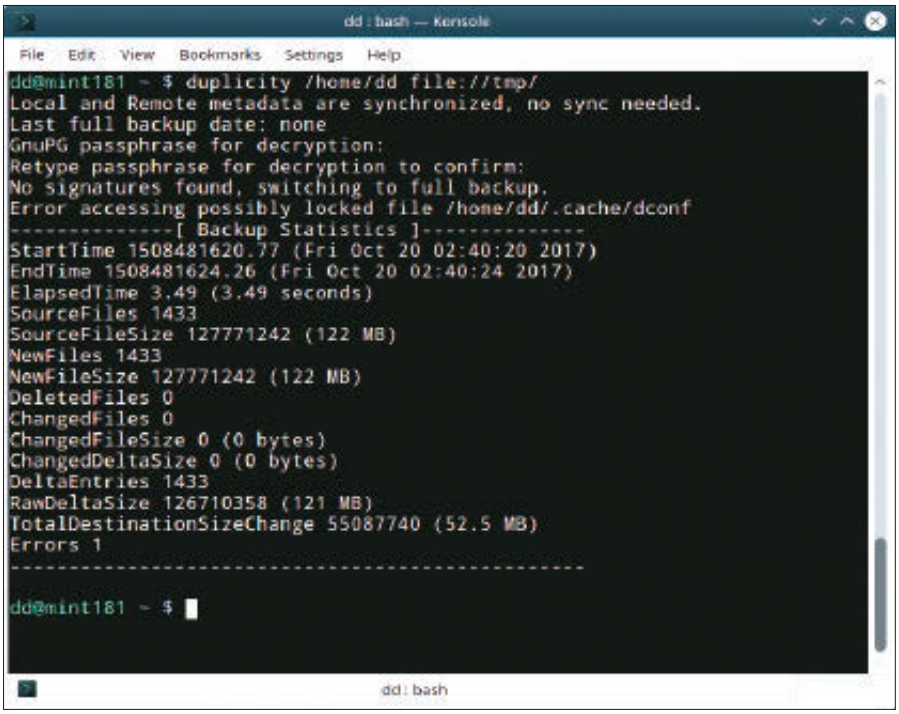
Przywracanie

Przywracanie danych jest równie łatwe – uruchamiamy Duplicity bez żadnych parametrów poza ścieżką źródłową i docelową, które teraz będą zamienione miejscami. Można ewentualnie określić jakąś inną ścieżkę docelową. Archiwa rozpakują się dopiero po wprowadzeniu hasła, aby nieupoważnione osoby nie miały dostępu do danych.

Parametrami *verify* oraz *--compare-data* łatwo sprawdzimy spójność naszych kopii, nawet gdy zbiory znajdują się w chmurze. Zalecany jest jeszcze parametr *-v<x>*, gdzie *<x>* to cyfra 4, 8 lub 9 – uzyskamy wtedy czytelne informacje przydatne zwłaszcza do dokumentacji kopii zapasowych. Dane te trafiają do pliku określonego parametrem *--log-file <nazwa-pliku>*.

rdiff-backup

Rdiff-backup [8] to kolejne popularne konsolowe narzędzie do kopii zapasowych. Program dostępny jest



Rysunek 6: Duplicity

w repozytoriach praktycznie wszystkich dużych dystrybucji Linuksa i zachwyca łatwością obsługi. Kopię lokalną katalogu lub drzewa wykonamy, wpisując:

```
rdiff-backup <katalog-źródłowy>
 <katalog-docelowy>
```

Powstanie kopia pełna, do odzyskania której administrator nie musi mieć żadnych narzędzi. Wystarczą operacje podobne jak w przypadku zwykłej linuksowej hierarchii folderów.

W kolejnych krokach do kopii rdiff-backupa trafiają pliki usunięte lub ich starsze wersje. Najnowsze wersje zmienionych plików lub nowe pliki są natomiast w kopii pełnej, dlatego będzie ona zawsze odzwierciedlać najnowszy stan danych (kopia utworzona zwykłym sposobem trzyma starszą wersję bazy). Jest to szczególna forma kopii przyrostowej o nazwie Reverse Delta, zwalniająca nas z konieczności przeczesywania kopii przyrostowych, gdy przywracamy całą bazę.

Serwery

Program Rdiff-backup wykona też kopie zapasowe serwerów w intranecie, a nie musi nawet być zainstalowany na serwerze. Aby stworzyć kopię zapasową przez SSH, po znaku zachęty zdalnego klienta wpisujemy:

```
rdiff-backup <użytkownik><ip-serwera>::
 <katalog-źródłowy>
 <katalog-kopii>
```

Na lokalnym kliencie powstanie kopia katalogu źródłowego serwera. Jeśli w kopii znajdują się pliki, do których dostęp ma tylko root, będzie trzeba zalogować się z odpowiednimi uprawnieniami.

Kwestia prezentacji

Podobnie jak inne testowane programy, Rdiff-backup dysponuje parametrami do zarządzania kopiami. Aby zobaczyć zawartość kopii, wpisujemy:

```
rdiff-backup -l <nazwa-kopii>
```

Rdiff-backup nie wyświetla żadnych komunikatów o stanie wykonywanej właśnie kopii, ale możemy przełączyć się na najwyższą szczegółowość parametrem *v6*. Program informuje wtedy o każdym przetwarzanym pliku, a po skończonym procesie pojawiają się komunikaty o załączonych i nieaktywnych parametrach (Rysunek 7).

Z parametrów *--compare* i *--list-increments* dostaniemy informacje o zmodyfikowanych plikach i zachowanych punktach kopiowania. Poleceniem:

```
rdiff-backup statistics
 <katalog-kopii>
```

wyświetlimy dane statystyczne o katalogu z kopiami (Rysunek 8).

Wyjątki

Podobnie jak w innych programach do kopii zapasowych, w backup-rdiff też można wykluczyć z kopiowania wybrane pliki lub katalogi, gdy nie chcemy zachowywać na przykład plików

```
linux-dq3w:~ # rdiff-backup -v6 /home/erik/Pictures/ /home/erik/rdiff-backup/
Using rdiff-backup version 1.2.8
Making directory /home/erik/rdiff-backup/rdiff-backup-data
Unable to import module posix1e from pylibacl package.
POSIX ACLs not supported on filesystem at /home/erik/Pictures
Unable to import win32security module. Windows ACLs
not supported by filesystem at /home/erik/Pictures
escape_dos_devices not required by filesystem at /home/erik/Pictures
-------------------------------------------------------------
Detected abilities for source (read only) file system:
  Access control lists                    Off
  Extended attributes                     On
  Windows access control lists            Off
  Case sensitivity                        On
  Escape DOS devices                      Off
  Escape trailing spaces                  Off
  Mac OS X style resource forks           Off
  Mac OS X Finder information             Off
-------------------------------------------------------------
Making directory /home/erik/rdiff-backup/rdiff-backup-data/rdiff-backup.tmp.0
Making directory /home/erik/rdiff-backup/rdiff-backup-data/rdiff-backup.tmp.0/hl
Hard linking /home/erik/rdiff-backup/rdiff-backup-data/rdiff-backup.tmp.0/hl/hardlinked_file2 to /home/erik/rdiff-
backup/rdiff-backup-data/rdiff-backup.tmp.0/hardlinked_file1
Unable to import module posix1e from pylibacl package.
POSIX ACLs not supported on filesystem at /home/erik/rdiff-backup/rdiff-backup-data/rdiff-backup.tmp.0
Unable to import win32security module. Windows ACLs
not supported by filesystem at /home/erik/rdiff-backup/rdiff-backup-data/rdiff-backup.tmp.0
escape_dos_devices not required by filesystem at /home/erik/rdiff-backup/rdiff-backup-data/rdiff-backup.tmp.0
Removing directory /home/erik/rdiff-backup/rdiff-backup-data/rdiff-backup.tmp.0
-------------------------------------------------------------
Detected abilities for destination (read/write) file system:
  Ownership changing                      On
  Hard linking                            On
  fsync() directories                     On
  Directory inc permissions               On
  High-bit permissions                    On
  Symlink permissions                     Off
  Extended filenames                      On
  Windows reserved filenames              Off
```

Rysunek 7: Rdiff-backup podaje znacznie więcej informacji niż inne programy do kopii zapasowych, ale trzeba umieć go poprosić.

tymczasowych. W takim przypadku wykluczamy pojedyncze pliki parametrem *--exclude*.

Jeśli w kopii mają zostać pominięte całe katalogi, korzystamy z parametru *--exclude-filelist*, pod którym wskazujemy plik ze ścieżkami do pomijanych plików. Taki plik trzeba stworzyć ręcznie.

Przywracanie

Zabezpieczone pliki da się przywrócić, nie wywołując ponownie Rdiff-backupa. Jeśli chcemy przywrócić dane z aktualnego archiwum, to zwyczajnie kopiujemy je ze ścieżki linuksowym poleceniem *cp*, wskazując jedynie parametrem, że chodzi o archiwum. W najprostszej postaci wygląda to tak:

```
cp -a /<katalog-kopii>/
  <nazwa-pliku> /<katalog-przywracania>/
  <nazwa-pliku>
```

Do starszych plików kopii można dostać się na dwa sposoby: przeglądać kopię przyrostową z programu rdiff-backup lub poleceniem *rdiff-backup-fs* zamontować archiwum z kopią jak zwykły system plików. Narzędzie *rdiff-backup-fs* to zewnętrzny pakiet do doinstalowania.

Narzędzie *rdiff-backup-fs* [9] (nazwa pakietu bywa też inna) figuruje w niektórych repozytoriach głównych dystrybucji Linuksa, ale można je też pobrać z witryny projektu. Bezpośrednio do kopii dostaniemy się za pomocą parametru *-r-* z datą (skrót od „restrore as-of”, czyli „przywróć do stanu z dnia x”). W przykładzie przywrócimy kopię sprzed 10 dni, z serwera do lokalnego katalogu klienta:

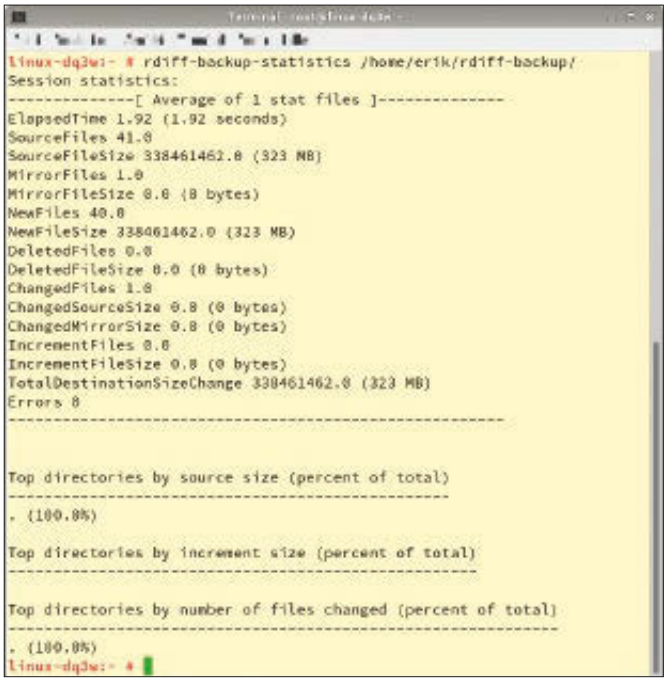
```
rdiff-backup -r 10D
  <Server Name>::/<Source Directory>/
  <Restore-Directory>
```

W szczegółowej dokumentacji [10] znajdziemy różne scenariusze zastosowania Rdiff-backupa, dzięki którym zrozumiemy momentami dziwaczną składnię programu.

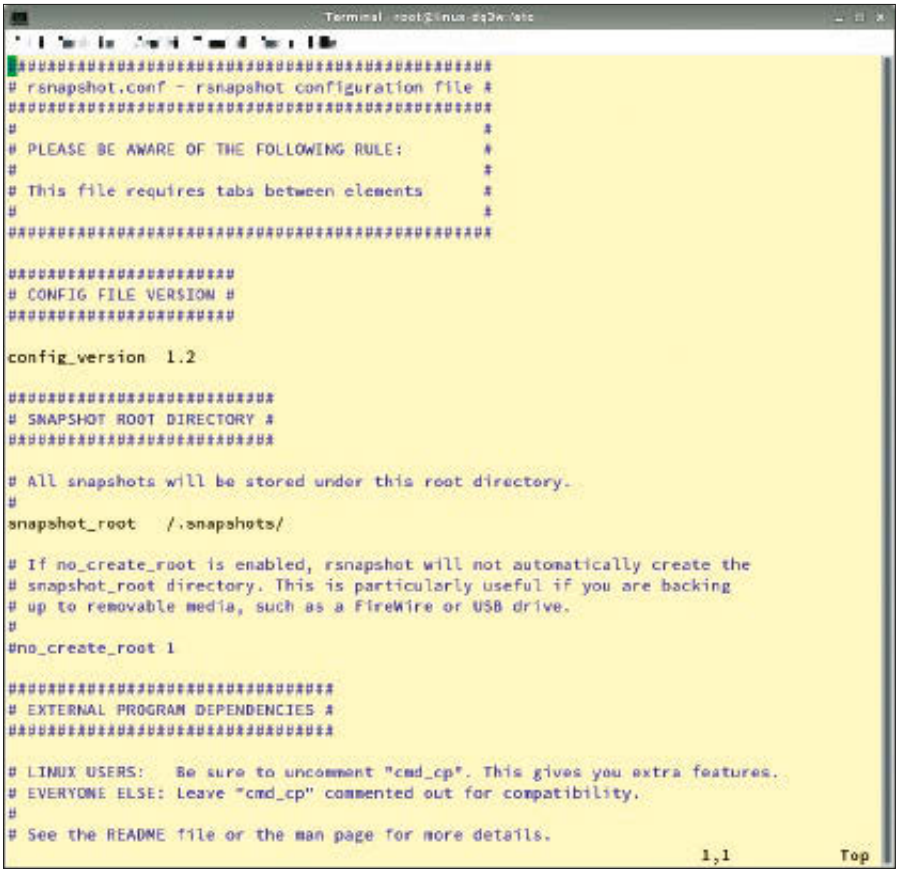
Rsnapshot

Jak sama nazwa wskazuje, Rsnapshot to narzędzie do tworzenia kompletnych migawek systemu plików [11]. Potrafi tworzyć zarówno migawki systemów lokalnych, jak i zdalnych serwerów po SSH. Narzędzie Rsync tworzy kopie, w których pliki bez zmian figurują jako bezwzględne dowiązania, po czym Rsnapshot nie zapisuje w kopii już niczego oprócz zmodyfikowanych danych. Proces tworzenia kopii jest inicjowany w regularnych odstępach przez Crona, a program pobiera potrzebne informacje z pliku konfiguracyjnego.

Rsnapshot korzysta z twardych dowiązań, więc kopie lądują zawsze w tym samym systemie plików – w przeciwnym wypadku trzeba by było tworzyć pełną kopię zajmującą wiele miejsca. Z tego powodu Rsnapshot lepiej nadaje się na serwery w niewielkich sieciach i stanowiska robocze niż do dużych magazynów danych. Liczbę migawek przechowywanych przez program można skonfigurować, więc zarządzanie miejscem nie stanowi problemu nawet przy częstych kopiach. Rsnapshot to narzędzie napisane w Perlu.



Rysunek 8: Jeden z wielu widoków na dane statystyczne ułatwiających sprawdzanie kopii.



Rysunek 9: Rsnapshot konfiguruje się za pomocą pliku.

Konfiguracja

Po instalacji programu, dostępnego w repozytoriach wszystkich popularnych dystrybucji Linuksa, skonfigurujemy go za pomocą pliku */etc/rsnapshot.conf* w dowolnym edytorze tekstu.

Szczegółowe komentarze pozwolą nam odnaleźć się w mnóstwie opcji, więc nawet dla niedoświadczonych użytkowników nie będzie tu problemów nie do rozwiązania. Pamiętajmy, że opcje rozdzielamy znakiem tabulatora, a definicje ścieżek muszą kończyć się ukośnikiem.

W plikach konfiguracyjnych określamy katalogi źródłowy i docelowy czy listę dołączonych/wykluczonych plików, ale można też tam zaplanować kalendarz regularnych kopii i kazać programowi kopiować zdalny serwer po SSH lub zawartość z sieci LVM. Ważną sprawą jest ustawienie czasu przechowywania stworzonych migawek (Rysunek 9).

Plik konfiguracyjny Rsnapshota jest spory, dlatego po zmianach dobrze jest go zweryfikować. Z uprawnieniami administracyjnymi wpisujemy *rsnapshot configtest*, a komunikat *Syntax OK* zasygnalizuje poprawną konfigurację. Następnie program zostanie automatycznie uruchomiony przez proces Crona lub Anacrona. W niektórych dystrybucjach dostępna jest predefiniowana konfiguracja na próbę, kryjąca się pod ścieżką */etc/cron.d/rsnapshot*, którą można dopasować do swoich potrzeb. Więcej informacji o programie znajdziemy w dziale FAQ [12].

Porządek dzięki rotacji

Rotacja danych pozwala dopasować liczbę tworzonych kopii do indywidualnych potrzeb. Odstępy między operacjami i liczbę kopii z każdego okresu można określić w pliku konfiguracyjnym. Parametr *retain* w */etc/rsnapshot.conf* przyjmuje różne określenia częstotliwości, np. *daily*, *hourly*, a po nim podajemy liczbę migawek.

Przykładowo *daily retain 5* oznacza uruchamianie kopiowania w odstępie doby, a zachowa się pięć migawek z codziennymi kopiami. W dalszych linijkach podobnie definiuje się więcej odstępów czasowych, przy których liczby mogą być inne, co czyni Rsnapshota bardzo elastycznym narzędziem.

Po zakończeniu konfiguracji możemy przetestować kopiowanie poleceniem:

```
rsnapshot -v <odstęp-czasowy>
```

W miejscu *<odstępu-czasowego>* wpisujemy określenie podane wcześniej w pliku konfiguracyjnym (np. *hourly*, *daily*). Nie trzeba określać katalogu źródłowego ani docelowego, ponieważ Rsnapshot pobiera te informacje z pliku.

Rsnapshot nie stosuje archiwów ani własnego formatu przechowywania plików, do danych mamy bezpośredni dostęp i możemy je przywrócić zwykłym skopiowaniem w pierwotne miejsce. Po udanym kopiowaniu testowym pozostanie już tylko ustawienie zadań dla Crona.

Wnioski

Wszystkie omówione tutaj rozwiązania do wykonywania kopii zapasowych działają stabilnie i wywiązują się z zadania – znajdą zastosowanie w kopiowaniu zarówno lokalnych systemów, jak i serwerów. Jeśli jednak chcemy trzymać archiwa kopii zapasowych w chmurze, muszą być one zaszyfrowane – tu większość programów odpadnie z eliminacji z powodu braku takiej funkcji.

Niektóre zaprezentowane programy mają także złą lub fatalną dokumentację – bywa, że dostępny opis nie uwzględnia parametrów funkcji ani przykładowych zastosowań, a strona podręcznika liczy raptem dziesięć linijek. W takiej sytuacji deweloperów nie powinna dziwić niewielka akceptacja i ograniczona cierpliwość ze strony użytkowników.

Opisane narzędzia powstały w większości na uniksowe systemy operacyjne i miewają wady projektowe, które mogą uniemożliwić działanie w środowiskach mieszanych – na przykład nie na każdej platformie czy systemie plików można korzystać z twardych dowiązań. W przypadku Rsnapshota ten czynnik ogranicza jego wykorzystanie tylko do Linuksa i spokrewnionych systemów. ■■■

INFO

[1] Attic: https://attic-backup.org/installation.html#installation

[2] bup na GitHubie: https://github.com/bup/bup

[3] Dokumentacja bupa: https://bup.github.io/man.html

[4] Duplicity: http://duplicity.nongnu.org

[5] Dokumentacja do Duplicity: http://duplicity.nongnu.org/duplicity.1.html

[6] Déjà Dup: https://launchpad.net/deja-dup

[7] Clonezilla: http://clonezilla.org

[8] Rdiff-backup: http://www.nongnu.org/rdiff-backup/

[9] Rdiff-backup-fs:https://github.com/rbrito/rdiff-backup-fs

[10] Dokumentacja do rdiff-backupa: http://www.nongnu.org/rdiff-backup/examples.html

[11] Rsnapshot: http://rsnapshot.org

[12] FAQ do Rsnapshota: http://rsnapshot.org/faq.html

Stacer: Wygodne czyszczenie systemu

Usługi porządkowe

Stacer to poręczne narzędzie z GUI, które pomaga w porządkowaniu systemu linuksowego. Ferdinand Thommes

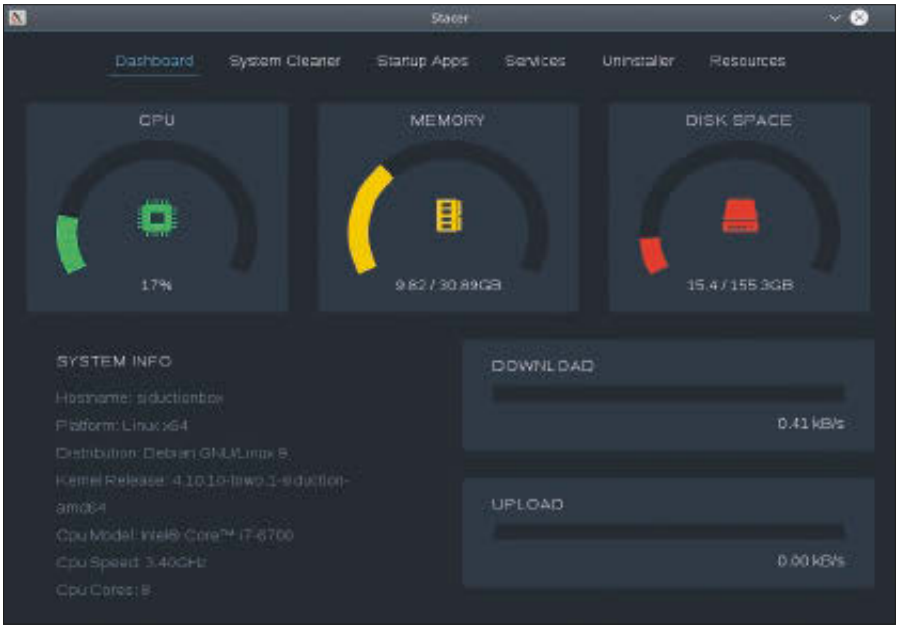
Zazwyczaj administratorzy korzystają z narzędzi działających w wierszu poleceń. Ma to praktyczne uzasadnienie: na serwerach najczęściej nie instaluje się środowiska graficznego, by nie dostarczać dodatkowych wektorów ataku. Istnieją jednak bardzo przydatne narzędzia administracyjne, które przydają się w monitorowaniu i optymalizacji systemu linuksowego – również w domu czy w firmie. Jednym z takich narzędzi jest Stacer, którego znajdziemy na GitHubie [1] i Sourceforge [2]. Do dyspozycji mamy źródła, pakiety DEB dla systemów 32- i 64-bitowych oraz AppImage dla maszyn 64-bitowych. W tym artykule przyjrzymy się wersji AppImage jako najbardziej uniwersalnej (patrz ramka AppImage).

APPIMAGE

Bardzo możliwe, że o AppImage [3] słyszymy po raz pierwszy. Istnieją bowiem inne systemy pakietów uniwersalnych, takie jak Snap czy Flatpak, które zyskały znacznie większą popularność. Choć AppImage rozwijany jest od 2004 r. (najpierw pod nazwą Klik [7], a później – PortableLinuxApps), pozostaje wciąż stosunkowo nieznany. Tak czy inaczej aplikacje dystrybuowane w formacie AppImage przypominają przenośne programy w Windows: można je uruchomić bez konieczności instalacji.

Czym jest Stacer?

Stacer został zaprojektowany z myślą o Ubuntu, lecz zadziała na praktycznie każdej dystrybucji. Aplikacja ta wykorzystuje stworzony przez programistów GitHuba framework Electron [4], za pomocą którego można budować wieloplatformowe aplikacje na bazie JavaScriptu, HTML-a i CSS. Popularne aplikacje korzystające z Electrona to Skype dla Linuksa, edytor Atom, komunikator Franz i narzędzie do przetwarzania obrazów Darktable.



Rysunek 1: Na ekranie startowym Stacera znajdziemy m.in. animowane kontrolki obrazujące obciążenie procesora oraz bieżące wykorzystanie pamięci i dysków.

Pobrawszy AppImage, rozpoczęliśmy testy na różnych maszynach wirtualnych: Ubuntu, Mint, Mageia, Manjaro, Apricity OS i openSUSE; spróbowaliśmy też statycznej instalacji w Siduction (Debianie *unstable*). Stacer zadziałał na wszystkich wymienionych dystrybucjach. Powinniśmy jedynie zwrócić uwagę, że w dystrybucjach z pulpitem KDE Plasma musimy uruchomić Stacera jako administrator ze względu na problem z autoryzacją *kdesu*.

Zanim rozpoczniemy pracę ze Staceremem, powinniśmy nadać AppImage prawa do wykonywania, co możemy zrobić w katalogu, w którym znajduje się pakiet. Następnie uruchamiamy aplikację z tego samego katalogu:

```
$ chmod a+x Stacer-1.0.6-x86_64.AppImage
$ ./Stacer-1.0.6-x86_64.AppImage
```

Pojawi się komunikat informujący, że aplikacja została dodana do menu i że na pulpicie pojawi się nowa ikona – to praktycznie jedyne zmiany wprowadzone na naszym komputerze. Stacera możemy też uruchomić, podobnie jak wszystkie inne aplikacje AppImage, klikając dwukrotnie ikonę pliku wykonywalnego.

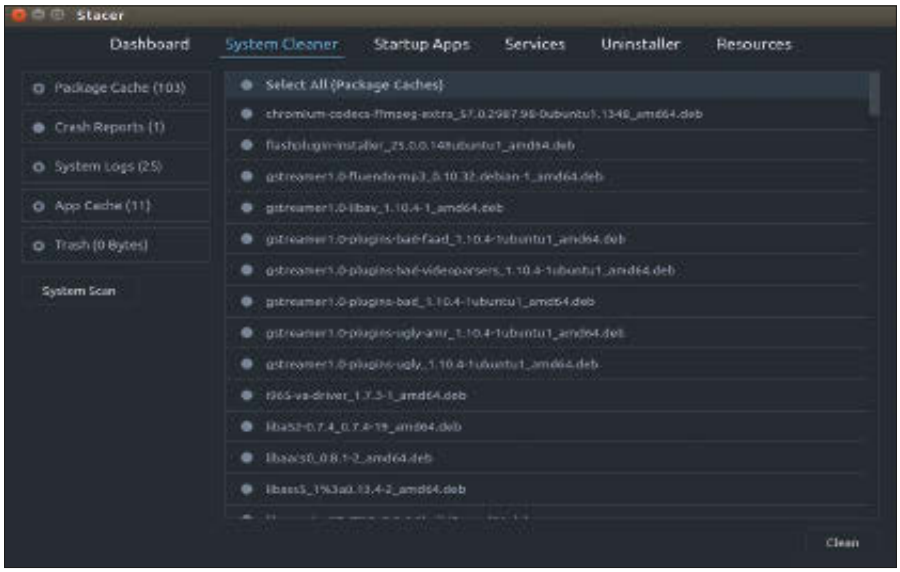
Graficzne zarządzanie

Po uruchomieniu Stacera ujrzymy nowoczesne okno (Rysunek 1) z sześcioma zakładkami: *Dashboard* (panel główny), *System Cleaner* (porządkowanie systemu), *Startup Apps* (aplikacje uruchamiane wraz z systemem), *Services* (usługi), *Uninstaller* (deinstalacja) oraz *Resources* (zasoby). Okno jest statyczne, tj. nie możemy zmienić jego wymiarów.

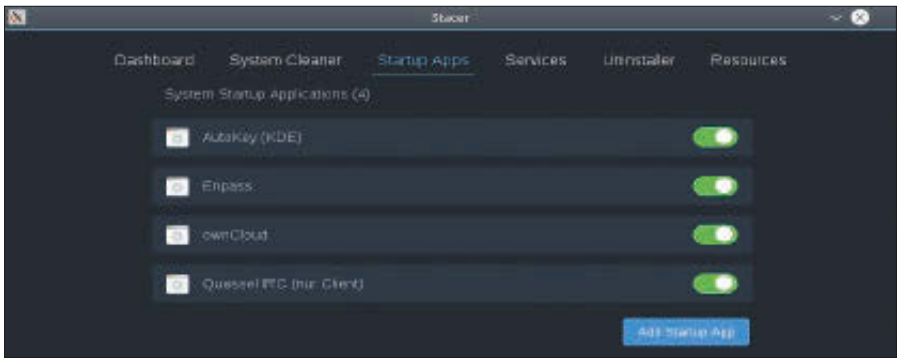
Po uruchomieniu programu zawsze wyświetlany jest panel główny, który służy jedynie do wyświetlania informacji i nie pozwala wprowadzać zmian. Znajdziemy tu animowany podgląd obciążenia procesora oraz bieżące wykorzystanie pamięci, dysków i interfejsów sieciowych, a także informacje dotyczące zainstalowanych procesorów i systemu operacyjnego.

Porządkowanie systemu

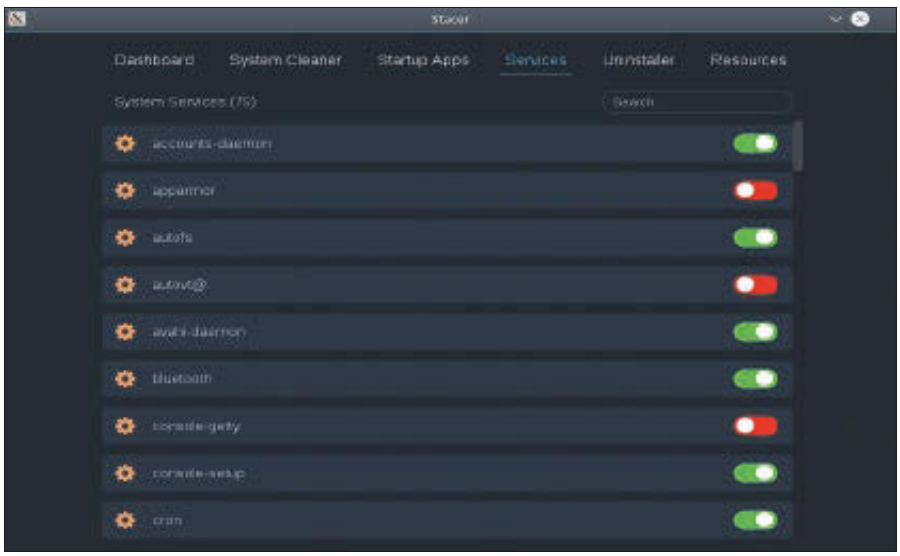
W zakładce *System Cleaner* (Rysunek 2) znajdziemy narzędzia umożliwiające pozbycie się balastu: w tym miejscu możemy usunąć zbędne pliki dziennika i pamięci podręcznej, a także opróżnić kosz. Na początku w koszu nie znajdziemy



Rysunek 2: W sekcji *System Cleaner* możemy pozbyć się tymczasowych plików, pozostałości po pamięci podręcznej oraz starych plików dziennika.



Rysunek 3: W sekcji *Startup Apps* możemy dodawać i usuwać aplikacje, które są uruchamiane przy starcie systemu.



Rysunek 4: Ostrożnie z uruchamianiem i zatrzymywaniem usług: błąd może zagrozić stabilności systemu.

żadnych danych: musimy najpierw włączyć poszczególne kategorie plików i uruchomić skanowanie systemu.

Korzystając z zakładki *App Cache*, powinniśmy zachować szczególną ostrożność: usunięcie niektórych plików może spowolnić działanie aplikacji. Nie warto też zbyt pochopnie usuwać plików dziennika – warto zachować choćby bieżące rejestry X, Apta i Dpkg. Pliki

dziennika oznaczone liczbami są starsze i często można się ich pozbyć bez negatywnych konsekwencji dla systemu.

Aplikacje i usługi

W zakładce *Startup Apps* możemy podejrzeć aplikacje uruchamiane wraz z systemem operacyjnym (Rysunek 3) i zarządzać nimi. Jest to przydatne zwłaszcza wtedy, gdy pracujemy z różnymi dystrybucjami: nie musimy się zastanawiać, w jaki sposób skonfigurować daną aplikację, by uruchamiała się przy starcie systemu. Możemy też nakazać Stacerowi, by uruchomił daną aplikację tylko przy najbliższym restarcie systemu.

Uruchamianie i zatrzymywanie usług jest równie proste – służy do tego zakładka *Services* (Rysunek 4). Dzięki funkcji wyszukiwania ławo znajdziemy żądaną usługę. Ostrożnie: jeśli zamkniemy niewłaściwą usługę, możemy zdestabilizować pracę systemu.

Do widzenia!

Ostatnia zakładka, podobnie jak pierwsza, pełni funkcję informacyjną. Z kolei przedostatnia, *Uninstaller*, umożliwia usuwanie pakietów (Rysunek 5). Znajdziemy tu wiele aplikacji zainstalowanych w systemie, które możemy odinstalować jednym kliknięciem. Stacer nie wyświetla tu głównych pakietów wymaganych przez system, by użytkownicy nie zrobili sobie krzywdy.

W naszych testach deinstalator zadziałał bez problemu jedynie na Ubuntu i Apricity. Przeglądając raporty o błędach na GitHubie natrafiliśmy na informację, że Stacer obsługuje tę funkcję jedynie na Ubuntu i Archu (na którym bazuje Apricity OS). Nie jest to jednak ogromny problem, ponieważ tak czy inaczej do zarządzania pakietami lepiej nadaje się menedżer pakietów wbudowany w daną dystrybucję.

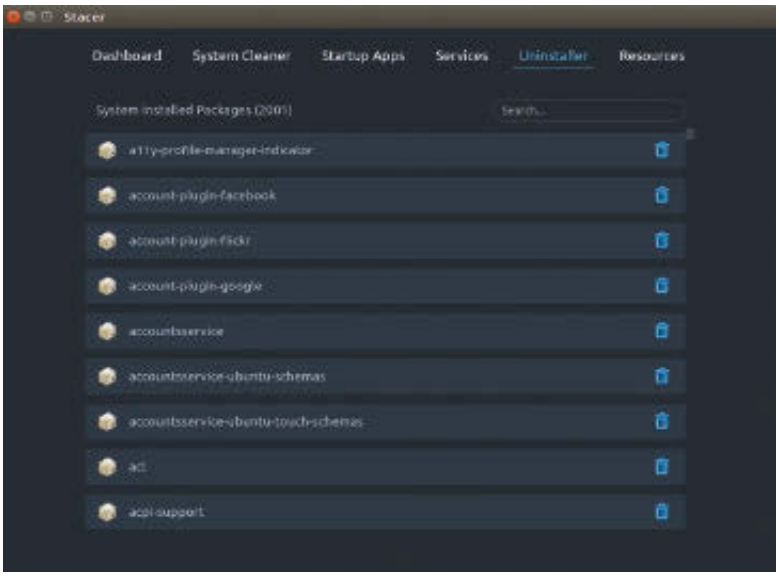
Kolorowych jarmarków

W zakładce *Resources* wyświetlanych jest 30 ostatnich sekund aktywności procesora, wykorzystania RAM-u i aktywności sieciowe (Rysunek 6). Jeśli mamy cztery i więcej rdzeni, Stacer wyświetli je w różnych, kontrastujących ze sobą kolorach. Aby prześledzić każdy wątek oddzielnie, klikamy przycisk *CPU History*.

Wnioski

Stacer nie jest niezbędnym narzędziem – można się bez niego obejść, korzystając z narzędzi uruchamianych w wierszu poleceń, standardowych komponentów danego środowiska graficznego czy konkurencyjnych projektów, takich jak BleachBit [5] – Stacer jest jednak wyjątkowo estetycznym rozwiązaniem, które z łatwością obsłuży każdy użytkownik. W naszych testach program najlepiej prezentował się na Apricity [6], ponieważ doskonale pasował do nowoczesnego wyglądu tej opartej na Archu dystrybucji.

Dzięki AppImage możemy korzystać ze Stacera na niemal dowolnej dystrybucji. Minusem jest rozmiar pakietu: AppImage waży ponad 50 MB; sam plik wykonywalny ma po rozpakowaniu ok. 75 MB, natomiast cały pakiet – 130 MB. Pamiętajmy też, że używając Stacera w niewłaściwy sposób, możemy narobić szkód w systemie. ■■■



Rysunek 5: Sekcja *Uninstaller* działa na Ubuntu, Archu i ich pochodnych.

Rysunek 6: W sekcji *Resources* możemy otworzyć wybrane obiekty (np. procesor) i oddzielnie śledzić obciążenie poszczególnych rdzeni.

INFO

[1] Stacer: https://github.com/oguzhaninan/Stacer
[2] Pobieranie Stacera: https://sourceforge.net/projects/stacer/files/v1.0.6/
[3] AppImage: https://en.wikipedia.org/wiki/AppImage
[4] Electron: https://electron.atom.io
[5] BleachBit: https://www.bleachbit.org
[6] Apricity: http://www.apricityos.com
[7] Klik: https://pl.wikipedia.org/wiki/Klik

Uczymy maszyny stanów SI rozpoznawać sekwencje wartości

Po kolei

Jaka liczba nastąpi po 2, 5, 7, 10, 12? Mike sprawdza, czy przygotowywane przez psychologów „testy na inteligencję” da się złamać za pomocą nowoczesnych maszyn stanowych. By Mike Schilli

Sieci neuronowe są naprawdę dobre w rozpoznawaniu wzorców wśród danych pełnych szumów i przypisywaniu im jednoznacznych wartości. Jeśli kilkadziesiąt osób napisze własnym charakterem pisma literę A lub B, wyćwiczona sieć może z niemal stuprocentową pewnością zidentyfikować dany znak. Pomyślmy o systemach rozpoznawania wzorców, które identyfikują numery rejestracyjne przejeżdżających samochodów – czy to nie cud techniki? Wyłuskują one cyfry ze strumienia wideo, pozwalając sprawdzić, kto gdzie jedzie.

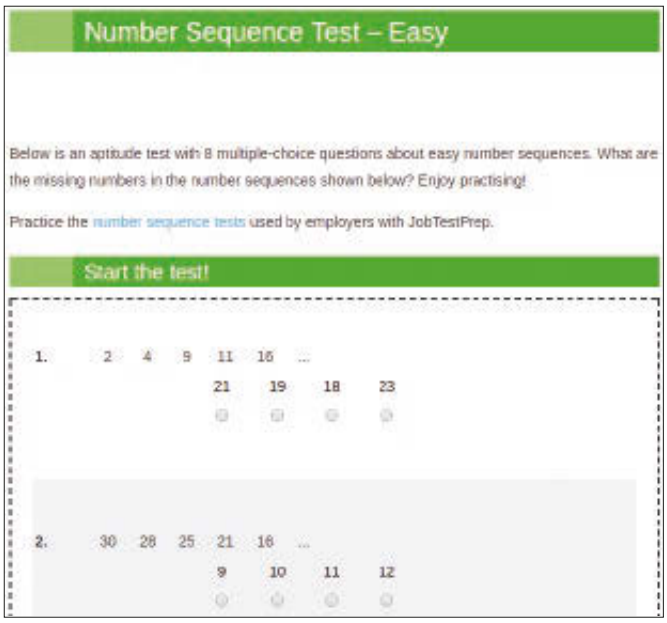
Kiedy sieć skończy naukę, będzie zawsze przypisywała ten sam wynik tym samym danym wejściowym. Jeśli jednak chodzi o określenie kolejnej wartości w sekwencjach dyskretnych czasowo, sieciom neuronowym często nie udaje się uzyskać doskonałych rezultatów, zwłaszcza jeśli sygnał wejściowy podlega zmianom o niejednolitej okresowości.

Algorytm uczący się w sieci neuronowej dopasowuje wewnętrzne wagi na podstawie danych, na których się uczy. Kiedy jednak wagi te zostaną ustalone, nie będą się zmieniały podczas działania programu, uniemożliwiając wprowadzenie korekt związanych z tymczasowymi zmianami w danych wejściowych, ponieważ maszyna nie zapamiętuje żadnego stanu poprzedniego. Istnieją co prawda sieci RRN (recurrent neural networks), które utrzymują wewnętrzne połączenia związane z wejściem, a zatem wynik może mieć wpływ na kolejny wektor wejściowy, nie pomaga to jednak prostej sieci zidentyfikować tymczasowe wzorce, które rozpościerają się na wiele cykli.

U psychologa

Rozważmy przykład przeprowadzanych przez psychologów tak zwanych testów na inteligencję (Rysunek 1). Niektóre z tych testów polegają na tym, że kandydat musi określić kolejną liczbę w ciągu. Każde dziecko wie, że po 2, 4, 6 pojawi się 8, ale jaka liczba wystąpi po 2, 5, 7, 10, 12?

Na Rysunku 2 widzimy dwa etapy nauki oraz krok testowy sieci „długiej pamięci krótkoterminowej” (Long Short-Term Memory, LSTM), którą chcemy nauczyć, jaka liczba wystąpi po 12. Na



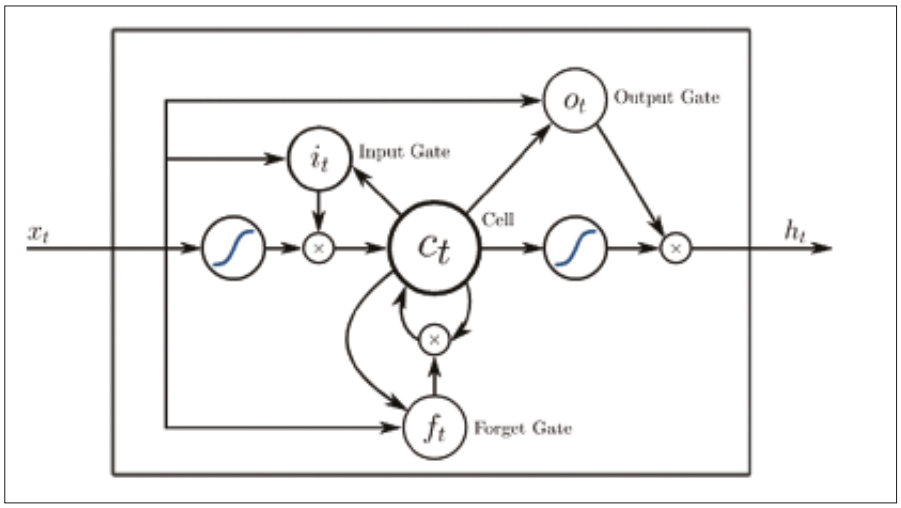
Rysunek 1: Test na inteligencję wymaga od badanego uzupełnienia ciągu liczb [1].

X1	X2	X3	Y
2	5	7	10
5	7	10	12
7	10	12	?

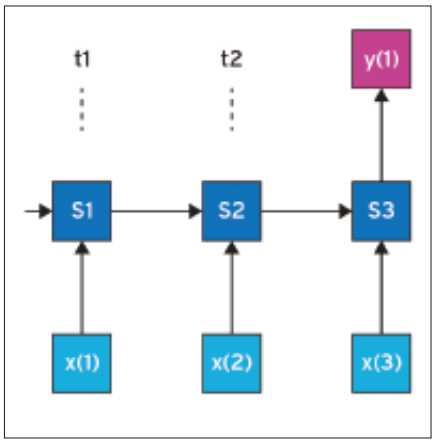
Rysunek 2: Wartości wejściowe i wyjściowe w procesie uczenia się sieci LSTM.

AUTOR

Mike Schilli pracuje jako programista w rejonie zatoki San Francisco. Można się z nim skontaktować, pisząc na: mschilli@perlmeister.com. Jego witryna znajduje się pod adresem: http://perlmeister.com.



Rysunek 3: Struktura komórki sieci LSTM.



Rysunek 4: Tymczasowo następujące po sobie wartości wejściowe na początku zmieniają jedynie bieżący stan wewnętrzny i wytwarzają dane wyjściowy co trzeci krok.

pierwszym etapie sieć uczy się, że po kombinacji *2, 5, 7* zawsze następuje 12. W drugim rzędzie przypisujemy wynik 12 podsekwencji *5, 7, 10*, badając tym samym okno przesunięte o jeden krok. Sieć LSTM wykorzystuje więc wyuczone dane do adjustacji parametrów swoich komórek wewnętrznych (Rysunek 3).

W przeciwieństwie do sieci neuronowych nie każda wartość wejściowa produkuje wartość wyjściową; zamiast tego LSTM śledzi bieżący stan w ukrytej komórce pamięci (Rysunek 4). Dopiero po otrzymaniu trzeciego fragmentu z wejściem i ewaluacji przeniesionego stanu pamięci produkowana jest wartość wyjściowa (*y(1)*).

Zmiana rozmiaru macierzy

Do implementacji sieci LSTM (Listing 1 [2]) użyjemy pythonowej biblioteki Keras [3]. Jako że wiele funkcji zawartych w tej bibliotece oczekuje danych w postaci macierzy o różnych kształtach i rozmiarach, warto wcześniej zaznajomić się z funkcją *reshape()* udostępnianą

Listing 1: iq

```
01 #!/usr/bin/python3
02 import numpy as np
03 from sklearn.preprocessing \
04       import StandardScaler
05 from keras.models import Sequential
06 from keras.layers import Dense, Activation
07 from keras.layers import LSTM
08 import os
09
10 os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
11
12 def window(npa, n=2):
13    for start in range(npa.size-n+1):
14       yield npa[start:start+n:1]
15
16 input_size=3
17
18 seq= np.array(
19       [2,5,7,10,12]).astype('float64')
20
21 print("learn input: " + str(seq))
22
23 scaler = StandardScaler()
24 seq = seq.reshape(-1,1)
25 seq = scaler.fit_transform(seq)
26 seq = seq.reshape(-1)
27
28 X=np.array([])
29 y=np.array([])
30
31 for chunk in window(seq, n=input_size+1):
32    X=np.append(X, chunk[:-1])
33    y=np.append(y, chunk[-1])
34
35 X=X.reshape((-1,input_size,1))
36 y=y.reshape((-1,1))
37
38 model = Sequential()
39 model.add(LSTM(5,
40    input_shape=(input_size,1)))
41 model.add(Dense(1))
42 model.add(Activation("linear"))
43 model.compile(loss="mean_squared_error",
44       optimizer="rmsprop")
45 model.fit(X,y, epochs=500, verbose=0)
46
47 print("\nwyniki:")
48 for input in X:
49    input=input.reshape(1,input_size,1)
50    pred=model.predict(input)
51    print(scaler.inverse_transform(
52       input.reshape(-1,1)))
53    print(scaler.inverse_transform(
54       pred.reshape(-1,1)))
55
56 test = seq[-input_size::1]
57 print(scaler.inverse_transform(
58    test.reshape(-1,1)))
59 test=test.reshape(1,input_size,1)
60 y1=model.predict(test)
61 print(scaler.inverse_transform(
62    y1.reshape(-1,1)))
```

```
$ python3
...
>>> import numpy as np
>>> a=np.array([1,2,3,4])
>>> print(a.reshape(-1,1))
[[1]
 [2]
 [3]
 [4]]
>>> print(a.reshape(-1,2))
[[1 2]
 [3 4]]
```

Rysunek 5: Tablica NumPy przyjmuje różne kształty dzięki *reshape()*.

przez bibliotekę tablic NumPy. Za pomocą *reshape()* przekształcamy jednowymiarową tablicę NumPy (czyli wektor) na macierz o ustalonych wymiarach (Rysunek 5).

Pierwszy parametr przekazywany *reshape()* to liczba elementów w pierwszym wymiarze, potem liczba elementów w drugim wymiarze itd. Ponieważ liczbę elementów określa niejawnie liczba elementów pozostających po zdefiniowaniu głębszych wymiarów, często wyraża się ją jako *-1*. Następnie biblioteka wypełnia macierz tym, co zostanie.

Po wywołaniu z pojedynczym parametrem (*reshape(-1)*) metoda konwertuje zagnieżdżoną strukturę tabeli z powrotem do wektora jednowymiarowego.

Zliczanie gier

Po wywołaniu z argumentem w postaci macierzy *[3,4,5,6,7]* (Rysunek 6) skrypt z Listingu 1 stosunkowo dokładnie odtworzy kolejną liczbę w ciągu (7,84 zamiast 8). Listing 1 dzieli ciąg liczb za pomocą zdefiniowanej w wierszu 12 funkcji *window* na przesuwane okna o czterech elementach (*[3,4,5,6]*, *[4,5,6,7]*); pierwsze trzy elementy przechowywane są w wektorze wejściowym *X*, ostatni zaś – w wektorze wynikowym *y*.

Aby wewnętrzne adjustacje wag sieci LSTM trzymały się rozsądnych wartości, *StandardScaler* z biblioteki *sklearn* normalizuje oryginalne wartości wejściowe do niewielkich dodatnich i ujemnych liczb zmiennoprzecinkowych zbliżonych do zera. Dotyczy to zarówno wektora wejścia, jak i rezultatu, przy czym ten ostatni powinien zawierać poprawne wyniki wymagane przy monitorowanym uczeniu się. Następnie metoda *fit_transform()* przeprowadza procedurę skalowania i standaryzuje dane. Później, zanim dojdzie do wyświetlania wyników, *inverse_transform()* przemapuje dane z powrotem do pierwotnej skali, byśmy mogli je właściwie przeanalizować.

W wierszach 38–45 poszczególne warstwy sieci LSTM są ustawiane jeden na drugim. Najpierw dodana jest główna warstwa LSTM (wiersz 39) o pięciu neuronach wewnętrznych. Dalej mamy połączoną warstwę wyjściową typu *Dense* i funkcję *Activation*, która ustawia krzywą odpowiedzi używanych wewnętrznie neuronów na liniową (*linear*), ponieważ w ten sposób podczas testów udało się osiągnąć najlepsze rezultaty.

Start!

Następnie model zostaje przygotowany do przetwarzania dzięki *compile()*; funkcja ta określa też *mean_squared_error* (odchylenia od optymalnych wyników mierzone są według błędu średniokwadratowego) jako parametr uczenia się i konfiguruje *rmsprop* jako mechanizm optymalizujący algorytm, co jest często stosowane w sieciach neuronowych.

W wierszu 45 wywołujemy metodę *fit* modelu, przekazujemy mu dane, których się nauczyliśmy, a następnie określamy liczbę iteracji służących do nauki jako *epoch = 500*. W naszych testach mniejsze liczby miały negatywny wpływ na wyniki, natomiast większe wartości *epoch* nie zapewniały znacząco lepszych rezultatów, a proces uczenia się znacząco spowalniał przy stałej wartości funkcji *loss*.

Sekcja zaczynająca się od wiersza 47 Listingu 1 monitoruje przewidywane wartości za pomocą modelu w bieżącej fazie uczenia się, zarówno w odniesieniu do przykładowych danych, jak i niewidzianych wcześniej ciągów, które system musi odgadnąć.

```
$ iq
Using TensorFlow backend.
learn input: [ 3.  4.  5.  6.  7.]

results:
[[ 3.]
 [ 4.]
 [ 5.]]
[[ 5.99448109]]
[[ 4.]
 [ 5.]
 [ 6.]]
[[ 6.99459839]]
[[ 5.]
 [ 6.]
 [ 7.]]
[[ 7.84399414]]
```

Rysunek 6: Sieć LSTM stosunkowo dokładnie obsługuje proste ciągi liczb.

```
$ ./iq
Using TensorFlow backend.
learn input: [ 1.  2.  3.  1.  2.  3.]

results:
[[ 1.]
 [ 2.]
 [ 3.]]
[[ 1.07632685]]
[[ 2.]
 [ 3.]
 [ 1.]]
[[ 2.34028244]]
[[ 3.]
 [ 1.]
 [ 2.]]
[[ 2.58298612]]
[[ 1.]
 [ 2.]
 [ 3.]]
[[ 1.07632685]]
```

Rysunek 7: W ciągu cyklicznym algorytm mniej więcej poprawnie przewiduje kolejne wartości.

Jak widzimy na Rysunku 7, sieć się sprawdza, nawet w odniesieniu do danych cyklicznych (1, 2, 3, 1, 2, 3), bez względu na długość okresu sygnału. Jest to ogromna zaleta w stosunku do tradycyjnych sieci neuronowych, w których musimy uprzednio określić okresowość, by prognozowane wyniki odpowiadały oczekiwaniom.

Szybki jak ślimak

Aby móc zainstalować bibliotekę Keras, musimy wcześniej zainstalować kilka innych modułów Pythona:

```
pip3 install --user keras pandas
  tensorflow sklearn numpy
sudo apt-get install python-tk
```

```
$ ./iq
Using TensorFlow backend.
learn input: [  2.   5.   7.  10.  12.]

results:
[[ 2.]
 [ 5.]
 [ 7.]]
[[ 9.98580551]]
[[  5.]
 [  7.]
 [ 10.]]
[[ 11.98676968]]
[[  7.]
 [ 10.]
 [ 12.]]
[[ 13.77653599]]
```

Rysunek 8: Sieć LSTM nie wypadła zbyt dobrze w teście na inteligencję, ponieważ nie udało się jej zidentyfikować różnych wartości wzrostu.

Używany przez Kerasa TensorFlow nie jest byt szybki: uruchomienie programu i rozpoczęcie uczenia się zajęło mojemu pięcioletniemu pecetowi dobre 10 sekund.

Ostatecznie jednak sieć nie wypadła zbyt dobrze w testach na inteligencję (Rysunek 8). Jak można się domyślić, do liczb w ciągu 2, 5, 7, 10, 12 trzeba na zmianę dodawać 2 i 3. Ponieważ skok od 10 do 12 wynosi 2, trzeba dodać 3, by przewidzieć kolejną liczbę – właściwy wynik to 15. Sieć jednak najczęściej zmierzała w stronę 14, nie może więc (jeszcze) konkurować z ludzką inteligencją. ■■■

INFO

[1] Kolejna liczba ciągu: https://www.fibonicci.com/numerical-reasoning/number-sequences-test/easy/

[2] Listingi do artykułu: http://www.linux-magazin.de/static/listings/magazin/2017/10/snapshot/

[3] Jason Brownlee, *Long Short-Term Memory Networks with Python: Machine Learning Mastery*: https://machinelearningmastery.com/lstms-with-python/

Dostawcy, którzy chronią przed atakami DDoS

Pod **ochroną**

Chcąc obronić się przed atakami DDoS, właściciele witryn i serwisów internetowych często sięgają po ochronę gigantów Internetu takich jak Amazon, jednak istnieją również inne sposoby ochrony łącz inerneowych. Konstantin Agouros

Ataki DDoS (ang. Distributed Denial of Service – rozpowszechniona odmowa usługi) są plagą z następstwami porównywalnymi do oprogramowania szantażującego (ransomware). Jest to w istocie rodzaj szantażu w formie nakierowanego ataku lub wirtualnego wandalizmu, kiedy to liczna grupa napastników zalewa witrynę internetową napływem żądań, próbując ją wyłączyć. W przeszłości napastnicy często wykorzystywali ataki odbijające, które polegały na wysyłaniu przez napastnika kilku pakietów przy użyciu adresu IP ofiary jako nadawcy do serwerów, które następnie potwierdzały żądania długimi odpowiedziami. Jednak z powodu sfałszowania adresu IP ogromna liczba odpowiedzi trafiała na adres ofiary ataku.

Nawet firmy o łączności z Internetem w zakresie od 10 do 40 Gbps mogą być bezradne wobec ataków o przepustowości setek gigabitów na sekundę. Jeżeli wprowadzimy w wyszukiwarce słowa kluczowe „biggest DDoS" (ang. największe ataki DDoS), ujrzymy coraz to nowsze ataki tego typu na coraz większą skalę; obecnie maksymalnie wynoszą one około 1 Tbps. W dobie Internetu Rzeczy atakujący mogą obecnie wybierać platformy, które są o wiele łatwiejsze do wykorzystania; przykładowo za ostatnie ataki odpowiedzialne były zhakowane kamery monitoringu, lodówki i tanie routery. W tym artykule przyjrzymy się metodom, dostawcom i kosztom ochrony naszego łącza internetowego.

Rodzina DDoS

Możemy rozróżnić zaledwie trzy kategorie ataków DDoS. Opisana powyżej metoda znana jako floodowanie (od angielskiego flooding – zalewanie) polega na wysyłaniu przez dużą grupę komputerów wielu ogromnych pakietów danych do ofiary, przeciążając przepustowość lub obciążając infrastrukturę do pełnej przepływności poprzez nadmierną ilość pojedynczych pakietów.

Podczas ataków wolumetrycznych firewalle lub systemy zarządzania obciążeniem utrzymują stan dla każdego połączenia i wprowadzają te stany do tabeli jednocześnie. Jeżeli istnieje wiele prób połączenia, tabela zostanie szybko zapełniona; Linux potwierdza pełne tabele, przekazując wiadomość *nf_conntrack: table full, dropping packet* (tabela pełna, odrzucenie pakietu). Jeżeli klient spróbuje uzyskać dostęp do Twojej strony w tym stanie, jego żądanie zostanie odrzucone. Zdarzały się przypadki, w których atak o przepustowości 2 Mbps wystarczył, by wyłączyć sieć chronioną zaporą. Zalewanie żądaniem synchronizacji SYN-flooding robi to samo z tabelą połączeń TCP jądra. W trakcie trójstopniowego uzgadniania, podczas nawiązywania połączenia, napastnik wysyła jedynie pierwszy pakiet, co oznacza, że dla połączenia utworzono wpis. Gdy już tabela się zapełni, wszelkie kolejne próby połączenia otrzymają odpowiedź *Connection Refused* (odmowa połączenia).

Wreszcie, przy wymyślnych atakach na czułe punkty aplikacji ataki napastników nie muszą być nawet „rozproszone". W trakcie mojej pracy jako tester penetracyjny zdarzyło mi się natrafić na funkcję wyszukiwania na portalu internetowym, która została wdrożona w języku Java i uruchamiała dla każdego wyszukiwania wątek podrzędny liczony w gigabajtach. Skrypt o 10 zapytaniach na sekundę sparaliżował całą platformę.

Tworzenie ochron przed atakami DDoS (floodwall)

Zapora uważana jest za standardowe narzędzie chroniące sieci. Jednak, jako że wykorzystuje ona jedynie dane pojedynczego pakietu jako kryterium

Ilustracja główna © Igor Stevanovic, 123RF.com

do zezwolenia na połączenia lub jego odrzucenia – a nie liczbę i wielkość pakietów – można ją w prosty sposób obejść. Lepsze zapory sieciowe (takie jak w jądrze Linuksa) dają możliwość kontrolowania przepustowości na połączenie, jednak zajmuje się tym procesor, a nie sprzęt sieciowy.

Osobną klasą są filtry urządzeń podobne do firewalli, jednak z funkcją sprawdzania ilości pakietów (całkowitą i na połączenie). Dedykowane specjalizowane układy scalone (ASIC) lub bezpośrednio programowalne macierze bramek (FPGA) określają reguły na poziomie sprzętu, jednak urządzenia te są dość drogie. Jeżeli chcemy przechwytywać ataki o rozmiarach 150 Gbps, to będziemy potrzebowali sześciocyfrowej kwoty.

Niestety, metody te mało pomogą, jeśli obrońca sieci znajduje się na wąskim końcu połączenia. Zanim urządzenie zacznie filtrować, połączenie zostanie już zablokowane. Mimo że zastosowane środki chronią infrastrukturę, nie będzie można uzyskać dostępu z zewnątrz ani do wewnętrznej sieci, ani też do witryny internetowej firmy.

Zasadniczo ochrona musi zostać wdrożona w dwóch krokach: po pierwsze, musimy zidentyfikować sam atak, a po drugie, musimy przekierować ruch ataku. Ataki możemy wykryć, badając różne parametry:

- Liczbę i rozmiar pakietów przesyłanych na docelowy port lub adres IP.
- Liczbę i rozmiar pakietów w ramach jednego połączenia; ponieważ połączenie zazwyczaj składa się z jednego pakietu żądania i jednego pakietu odpowiedzi, 20 pakietów żądania już wygląda na podejrzane.
- Łączną sumę tych danych.

Zbieranie danych

Tylko nieliczne protokoły i aplikacje stosują się do statycznych zasad. O ile 100 żądań na godzinę to normalna ilość w przypadku sklepu internetowego, o tyle liczba ta może szybko się zwiększać w okresie przedświątecznym lub podczas kampanii marketingowych. Regularne zmiany w wartościach w odniesieniu do pory dnia lub dnia tygodnia są również powszechne. Programy statystyczne mogą wykryć, co jest normalne, i oznaczyć to w czasie. Jeżeli odchylenie od normy wzrasta, można to zakwalifikować jako atak.

Dwa podejścia pozwalają na zbieranie danych o ruchu: (1) instalacja urządzenia za pomocą wstawki (np. przy użyciu dwóch kabli), która działa jako mostek pomiędzy wnętrzem a tym co na zewnątrz. Urządzenie następnie rejestruje wszystkie dane w trybie swobodnym i tym samym zbiera statystyki. (2) Podłączenie takiego urządzenia do lustrzanego portu przełącznika, tak by awaria nie zakłóciła połączenia.

Inną opcją jest wykorzystanie NetFlow [1], Slow [2] lub Ipfix (eksport przepływu informacji protokołu internetowego) [3]. Protokoły te dostarczają poszukiwane dane połączenia (o różnych poziomach szczegółowości). Elementy sieci, takie jak routery lub przełączniki, wysyłają statystyki do odbiornika, który następnie przeprowadza analizę statystyczną i wysyła ostrzeżenia.

Przełączanie na tryb obronny

Na środki obronne składa się blokowanie niechcianego ruchu. Na ogół administrator odfiltrowuje atakowany docelowy adres IP (lub blok sieci) poprzez tworzenie list kontroli dostępu (ACL) na routerach lub poprzez określenie tzw. trasy zerowej za pomocą zewnętrznego protokołu trasowania (Border Gateway Protocol – BGP) [4]. Wpis ten w tabeli trasowania odrzuca wszystkie pakiety próbujące dotrzeć do atakowanego adresu IP na wlocie routera. W wyniku tego atakowany serwer pozostaje offline, ale reszta linii pozostaje wolna.

Rozszerzenie BGP flowspec [5], które pozwala administratorom na dystrybucję ACL-ów zawierających docelowe porty i protokoły poprzez protokół BGP, jest bardziej dokładne. Jeżeli atak odbijający NTP (Network Time Protocol – protokół czasu sieci) wysyła wiele pakietów do portu UDP 123 na serwerze, a router znajdujący się wyże tylko go blokuje, to strony trzecie mogą nadal dotrzeć do serwera poprzez porty 80 i 443 (TCP); jednak niewielu producentów routerów wspiera tę strategię.

Ogólnie rzecz biorąc, mało którzy dostawcy pozwalają klientom na rozprowadzanie zasad filtracji tego typu w swoich routerach. Jeżeli dostawca posiada potężne narzędzia do ochrony przed atakiem DDoS, będzie w stanie przekierować zainfekowany ruch i umożliwić wprowadzenie zasad dopasowania, tak by do ofiary docierały jedynie pożądane dane.

Środki obronne przed floodowaniem należy zastosować po szerszej stronie połączenia (tzn. po stronie dostawcy) lub w centrum danych, gdzie obsługiwane są Twoje strony. Rodzi to pytania o przepustowość łącza w centrum danych. Połączenie 40 Gbps może w zupełności wystarczyć do normalnego działania centrum danych, jednak skuteczny atak DDoS na nie będzie bardzo łatwy. Byłem świadkiem ataku o mocy 200 Gbps na pojedynczy serwer i nie był to atak na firmę komercyjną.

Czyszczenie danych

Poszczególni dostawcy oferują swoim klientom ochronę przed DDoS za opłatą. Niektórzy kupują drogi sprzęt, który może to zrobić, a następnie wypożyczają go klientom.

Najczęstszymi działaniami jest łagodzenie ataków w chmurze, podczas których dedykowany dostawca „czyści” ruch, uruchamiając farmę urządzeń anty-DDoS w centrum danych połączonym z siecią szerokopasmową, do którego ruch ofiary jest przekierowany w celu oczyszczania, a następnie przekierowany z powrotem do celu przez tunel.

Istnieją dwie opcje przekierowania ruchu. Z jednej strony można wykonać go na poziomie DNS. Jeżeli atak skierowany jest przeciwko *www.przyklad.pl*, możemy przekierować wpis DNS do adresu IP w systemie czyszczenia. Jednak odkrycie tej zmiany przez resztę Internetu zajmie trochę czasu. Administratorzy muszą utrzymywać liczbę żądań DNS na niskim poziomie, tak by buforowanie nazw serwerów nie opóźniało przekierowania.

Z drugiej strony protokoły routingu (na ogół są to BGP) mogą przekierowywać ruch. Zauważmy, że nie działa to przy poszczególnych adresach IP, a jedynie z pełnymi blokami sieci. Zaletą tego rozwiązania jest to, że te zmiany ścieżki rozprzestrzeniają się w sieci szybciej.

Wąskie gardło

Obrona przed atakami wolumetrycznymi może być również stosowana na wąskim końcu linii. Jeśli Netfilter jest używany jako zapora sieciowa, pierwszym krokiem będzie sprawdzenie parametru systemowego *net.netfilter.*

nf_conntrack_max (lub */proc/sys/net/netfilter/nf_conntrack_max*), który określa maksymalną liczbę połączeń. Wartość domyślna waha się pomiędzy *32768* a *65536*, które atakujący może wyczerpać w miarę szybko.

Można zastosować *sysctl -w*, ażeby zwiększyć tę wartość do 2 GB. Jednak urządzenie musi mieć wystarczającą pamięć fizyczną, aby zapisać tę liczbę wpisów. Wpis pochłania dobre 300 bajtów (w rzeczywistości jest nieco bardziej skomplikowany, a post na blogu [6] opisuje go dokładniej), co oznacza, że potrzebne jest ponad 700 GB pamięci RAM na 2 miliardy wpisów. Według cytowanej strony jądro może obsłużyć 1,7 miliona wpisów przy zarezerwowaniu około 512 MB na śledzenie połączenia, co jest o kilka rzędów wielkości większe niż wartość domyślna.

Jądro może używać plików cookie SYN do obrony przed SYN floodingiem. Serwer wysyła odpowiedź, ale nie tworzy wpisu w tabeli. W atakach SYN flooding brakuje trzeciego pakietu. Jeśli się pojawi, serwer rozpozna sytuację i utworzy wpis. Aby było to możliwe, jądro konstruuje numery sekwencji, ażeby je rozpoznać w pakiecie ACK.

Urządzenia DDoS również mogą to robić, ale oferują też możliwość pracy jako proxy TCP, co oznacza, że najpierw uzupełniają uzgadnianie działające jako proxy i otwierają połączenie z serwerem tylko wtedy, gdy on faktycznie działa. Większość nowoczesnych systemów firewall może również korzystać z tej metody.

Na tym samym poziomie istnieje również obrona przed silniejszymi atakami. W przypadku aplikacji internetowych odwrotne proxy, takie jak Apache z *mod_security*, może zapobiec naruszeniu bezpieczeństwa. Moduł umożliwia określenie zasad i ograniczenie liczby żądań na każdy źródłowy adres IP. Należy jednak skonfigurować to ręcznie dla każdego adresu URL.

Features	AWS Shield Standard	AWS Shield Advanced
Active monitoring		
Network flow monitoring	✔	✔
Automated application (layer 7) traffic monitoring	-	✔
DDoS mitigations		
Helps protect from common DDoS attacks, such as SYN floods and UDP reflection attacks	✔	✔
Access to additional DDoS mitigation capacity	-	✔
Visibility and reporting		
Layer 3/4 attack notification and attack forensic reports	-	✔
Layer 3/4/7 attack historical report	-	✔
DDoS response team support		
Incident management during high severity events	-	✔
Custom mitigations during attacks	-	✔
Post-attack analysis	-	✔
Cost protection		
Reimburse related Route 53, CloudFront, and ELB DDoS charges	-	✔

Rysunek 1: Amazon, w pewnym stopniu, chroni swoich klientów przed atakami DDoS. Jeśli potrzebujemy lepszej ochrony, będziemy musieli bardzo głęboko sięgnąć do kieszeni.

Urządzenia DDoS pozwalają na ustalanie limitów na poziomie transakcji, ale najpierw należy sprawdzić u producentów, które protokoły są przez nie obsługiwane. A10 Networks [7] oferuje nawet dynamiczne włączanie Captchas po przekroczeniu progu, a następnie zezwala na dalsze żądania do danego adresu źródłowego, o którym mowa, po tym, jak użytkownik potwierdził, że jest człowiekiem za pomocą Captcha.

Osłona AWS

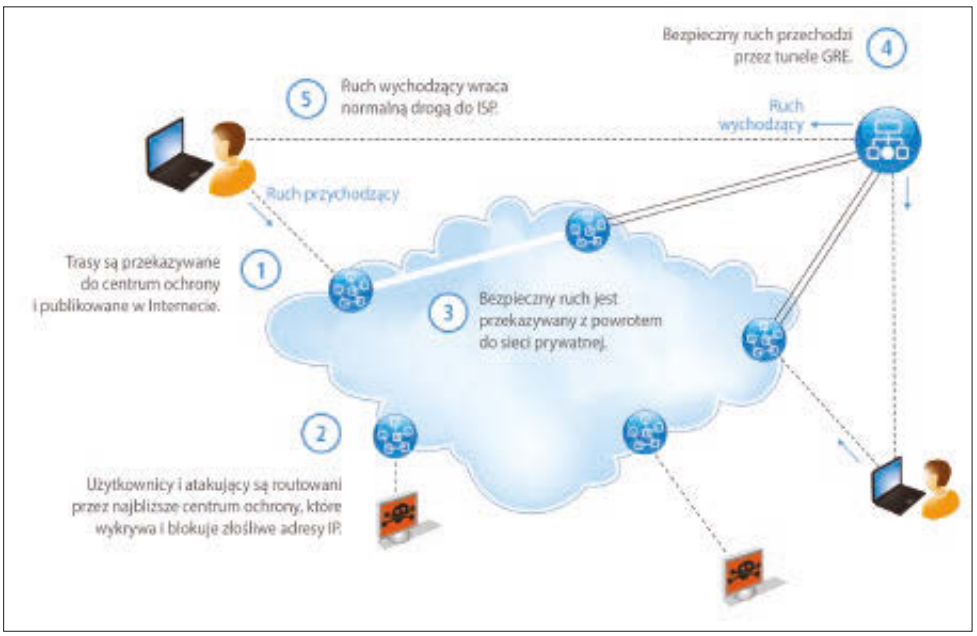
Osłona Amazon Web Services (AWS) [8] zapewnia ochronę przed atakami DDoS (rysunek 1). Standardowa ochrona jest dostępna dla każdego klienta AWS. Produkt obejmuje wykrywanie danych o przepływie sieciowym i automatyczne łagodzenie ataków DDoS przeciwko zalewaniu żądaniami synchronizacji SYN lub atakom odbijającym UDP. Niemniej nie otrzymasz informacji o udanej obronie. Jeśli wybierzemy produkt AWS Shield Advanced, otrzymamy następujące dodatkowe funkcje za około 3025 USD miesięcznie plus opłaty za przesył danych:

- Poza danymi dotyczącymi połączeń na poziomie sieci Amazon gromadzi i analizuje rejestry zdarzeń transakcji na poziomie aplikacji.
- Dostęp do zaawansowanych funkcji czyszczenia.
- Powiadomienie o atakach na warstwach ISO 3 i 4, a także informacje o rodzaju ataku.
- Raporty dla warstw ISO 3, 4 i 7.
- Zarządzanie incydentami przez zespół Amazon DDoS.
- W razie potrzeby ręczne łagodzenie.
- Ręczna analiza po ataku.
- Zwrot kosztów poniesionych w wyniku ataku związanego z usługami CloudFront, Route 53 i ELB.

Warto zauważyć, że Amazon chroni tylko to, co działa w serwisie Amazon. Mimo że możliwe jest zabezpieczenie ruchu danych na własnych serwerach za pomocą usług takich jak CloudFront lub odwrotnego serwera pośredniczącego, a także chronienie swoich połączeń sieciowych w inny sposób, nie możemy zwalczać nakierowanych ataków.

Urządzenia Arbor

Arbor [9] tworzy systemy wykrywania i obrony przed DDoS za pomocą własnego sprzętu. Firma oferuje systemy oczyszczania ruchu w centrach danych



Rysunek 2: Akamai, znany z globalnego CDN, dostarcza klientom płatną ochronę przed atakami.

w USA, Europie i Azji. Usługa chroni tylko przed atakami wykrytymi przez klienta.

Licencjonowanie zależy od przepustowości czystego ruchu. Jeśli klient ma linię o przepustowości 1 Gbps, to płaci za pakiet 1 Gb. Dodatkowo Arbor oferuje pakiety za miesięczną opłatę lub na obronę przed 12 atakami rocznie. Każdy pakiet zawiera ochronę sieci /24, w tym pięć domen DNS i można go rozszerzyć. Arbor pozwala nam połączyć uruchamianie mechanizmów obronnych z naszym urządzeniem. Po wykryciu ataku uruchamiana jest obrona chmurowa i dostarczane są raporty o atakach.

Link11

Link11 [10] również oferuje ochronę przez BGP, przez przekierowanie DNS i bezpośrednie połączenie z centrami danych dostawcy usług hostingowych. Licencje na wersje DNS i BGP różnią się, ale oba określają czysty ruch jako 95. centyl zwykłego ruchu (bez ataków).

W przypadku DNS Link11 zlicza, ile adresów IP jest przez niego chronionych w oryginalnych systemach, przy poziomach czystej szybkości ruchu wynoszącymi 25, 50, 100, 150 i 250 Mb/s. W przypadku BGP rozmiar maski sieciowej jest liczony jako parametr, a liczenie rozpoczyna się od sieci /24. Druga opcja to ponownie chroniona przepustowość (Link11 rozróżnia routing symetryczny i asymetryczny); dostępne skale to 250, 500 i 1000 Mb/s.

Akamai

Akamai [11] zarządza jedną z największych sieci dostarczania treści (CDN) na całym świecie i wykorzystuje swoją sieć do oferowania ochrony DDoS w chmurze. Oprócz wariantów DNS i BGP, Akamai sprzedaje ochronę proxy, która zapewnia ich ochronę naszych aplikacji (np. porty na adresach IP).

Podobnie jak Link11, produkt Prolexic Connect zapewnia centrom danych opcję ochrony pod skrzydłami CDN (rysunek 2). Podobnie jak Link11, Akamai ustaliło model rozliczeniowy jako subskrypcję i stosuje zasadę 95. percentylu czystego lub normalnego ruchu przychodzącego do obliczenia ceny. W wariancie BGP istotne są również liczba sieci /24 i liczba chronionych witryn.

Wnioski

Jeżeli obsługujemy swoją infrastrukturę z AWS, powinna wystarczyć nam podstawowa ochrona. Jednak zaawansowana ochrona jest dość droga. Zastanówmy się, czy korzystanie z usług Amazona jest tak istotne dla naszej firmy, że opłaca się płacić za nie ponad 3025 USD miesięcznie. Opcja zapłaty za atak może szybko mieć negatywny wpływ na grubość portfeli ofiar. Jeśli chcemy całkowicie uniknąć Amazona, musimy rozejrzeć się za innym dostawcą ochrony DDoS.

Informacje w tym artykule prawdopodobnie mogą pomóc obrać właściwy kierunek, jednak nie przedstawiają uniwersalnego rozwiązania. Jako że struktury cen u dostawców różnią się znacznie, należy najpierw dokładnie przeanalizować swoją sytuację pod względem zagrożeń. Czy już doświadczyliśmy ataków lub jesteś nimi zagrożeni? Jeśli tak, czy zagrożona była cała firma, czy tylko jeden dział? W tym drugim przypadku nawet Amazon może być przydatny, zwłaszcza jeśli natężenie ruchu nie jest zbyt wysokie.

Nawet jeśli nie podejmiemy żadnych środków zaradczych przeciwko atakom DDoS za pomocą określonych środków, powinniśmy przynajmniej mieć plan awaryjny w zapasie i wcześniej poznać swoje opcje wyboru. Link11 i Arbor w przypadku ataku oferują na swoich stronach internetowych linki awaryjne. W przypadku zagrożenia liczy się czas. ■■■

AUTOR

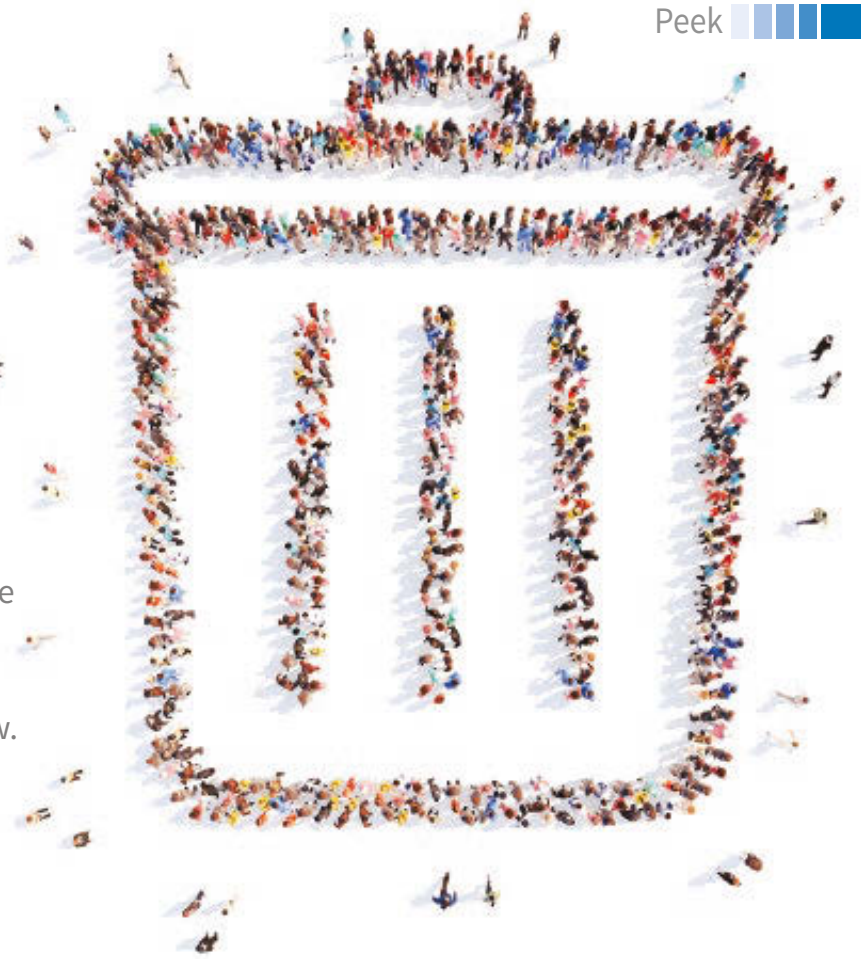
Konstantin Agouros pracuje jako kierownik projektów Open Source w firmie Matrix Technology AG, gdzie wraz z zespołem doradza klientom w zakresie oprogramowania open source i chmur. Jego nowa książka *Software Defined Networking: SDN-Praxis mit Controllern und OpenFlow* [Praktyka-SDN z urządzeniami sterującymi i OpenFlow] została opublikowana przez De Gruyter.

INFO

[1] NetFlow: https://en.wikipedia.org/wiki/Netflow
[2] sFlow: http://www.sflow.org
[3] Ipfix: https://tools.ietf.org/html/rfc7011
[4] Border Gateway Protocol: https://tools.ietf.org/html/rfc1772
[5] Flowspec: https://tools.ietf.org/html/rfc5575
[6] Wykorzystanie pamięci Netfiltra Conntrack: https://johnleach.co.uk/words/372/netfilter-conntrack-memory-usage
[7] A10: https://www.a10networks.com/products/thunder-series/ddos-detection-protection-mitigation
[8] AWS Shield: https://aws.amazon.com/en/shield/
[9] Arbor: https://www.arbornetworks.com/ddos-protection-products
[10] Link11: https://www.link11.com/en/
[11] Akamai: https://www.akamai.com/us/en/resources/ddos.jsp

Peek – nagrywanie screencastów w Gnome

Zerkanie przez ramię



Screencast to film pokazujący to, co dzieje się na ekranie komputera. Peek pozwala błyskawicznie tworzyć screencasty i eksportować je do popularnych formatów.
Christoph Langner

Wszyscy znamy banał o tym, że obraz wart jest więcej niż tysiąc słów. A ile słów opisu oszczędzimy dzięki pokazaniu czegoś na wideo? W wielu przypadkach krótki screencast, czyli zapis zdarzeń z naszego ekranu, lepiej wyjaśni, gdzie jest problem lub jak wykonać jakąś czynność, niż długi tekst z ilustracjami. W kwestii narzędzi do tworzenia screencastów naprawdę jest z czego wybierać.

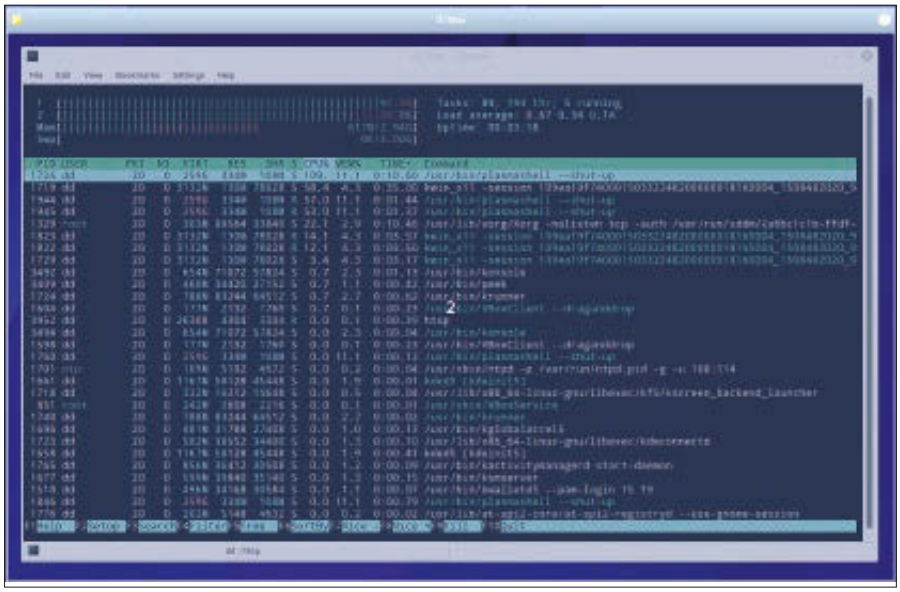
Film z pulpitu możemy nagrać programami takimi jak SimpleScreenRecorder [1] czy recordMyDesktop [2]. W porównaniu z tymi zawodnikami Peek [3] nie ma zbyt dużej funkcjonalności, ale i nie próbuje być konkurencją dla uznanych graczy. Pierwotnie służył do zapisywania zdarzeń z ekranu w formie gifów, dzięki czemu łatwiej było osadzić ruchomy obraz na stronach internetowych. Teraz obsługuje również bardziej tradycyjne formaty wideo – WebM i MP4.

Nagrywanie pulpitu

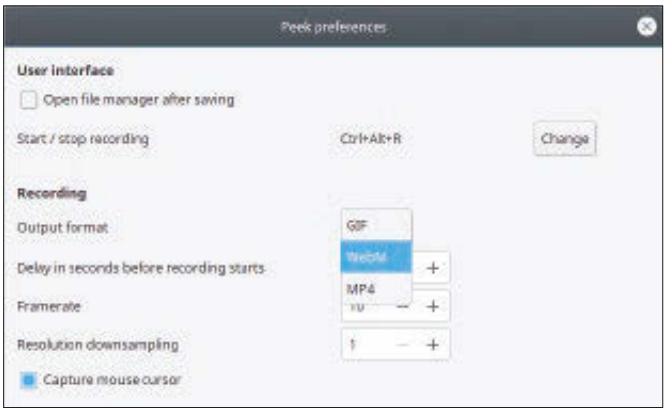
W kwestii interfejsu Peek przypomina windowsowe narzędzie LICEcap [4] i jest to zabieg celowy. Program wyświetla skalowalne przezroczyste okno, które jest zawsze na wierzchu wszystkich otwartych aplikacji. Po kliknięciu przycisku *Record* wszystko wewnątrz okna jest przechwytywane przez program jako wideo. Kiedy wciśniemy *Stop*, Peek od razu zapisuje materiał na dysku (Rysunek 1).

Prawdziwe formaty wideo (WebM i MP4) program obsługuje dopiero od wersji 1.0.0, opublikowanej w marcu 2017 r. Do opcji *Output format* dostaniemy się jak na Rysunku 2.

Podobnie jak we wszystkich nowych aplikacjach na Gnome, opcje formatów otwieramy z menu aplikacji obok menu czynności na górnej belce całego pulpitu. Teraz możemy ustawić także inne parametry, takie jak opóźnienie do rozpoczęcia nagrywania i częstotliwość odświeżania (*Framerate*), a także zmniejszyć rozdzielczość nagrywanego obszaru (*Resolution downsampling*).



Rysunek 1: Przed rozpoczęciem nagrywania program odlicza (konfigurowalny) czas, aby można było wszystko przygotować.



Rysunek 2: Docelowy format wideo wybierzemy pod *Peek preferences*. Oprócz formatu GIF film zapiszemy jako WebM i MP4.

Posługując się formatem GIF, trzeba pamiętać, że nie był on pierwotnie przeznaczony do nagrywania filmów – nagrywanie całego pulpitu w full HD przy 30 klatkach na sekundę będzie generowało olbrzymie pliki.

Do nagrania wybierajmy więc wycinek, w którym rzeczywiście znajduje się coś istotnego. Okienko można wyskalować co do piksela dzięki wyświetlanym wymiarom (Rysunek 3). Można ponadto zmniejszyć odświeżanie do ok. 10 fps, a w razie potrzeby również zmniejszyć rozdzielczość (*Resolution downsampling*) o zadaną liczbę całkowitą.

Listing 1: Instalacja z PPA

```
sudo add-apt-repository ppa:peek-developers/stable
sudo apt update
sudo apt install peek
```

Instalacja

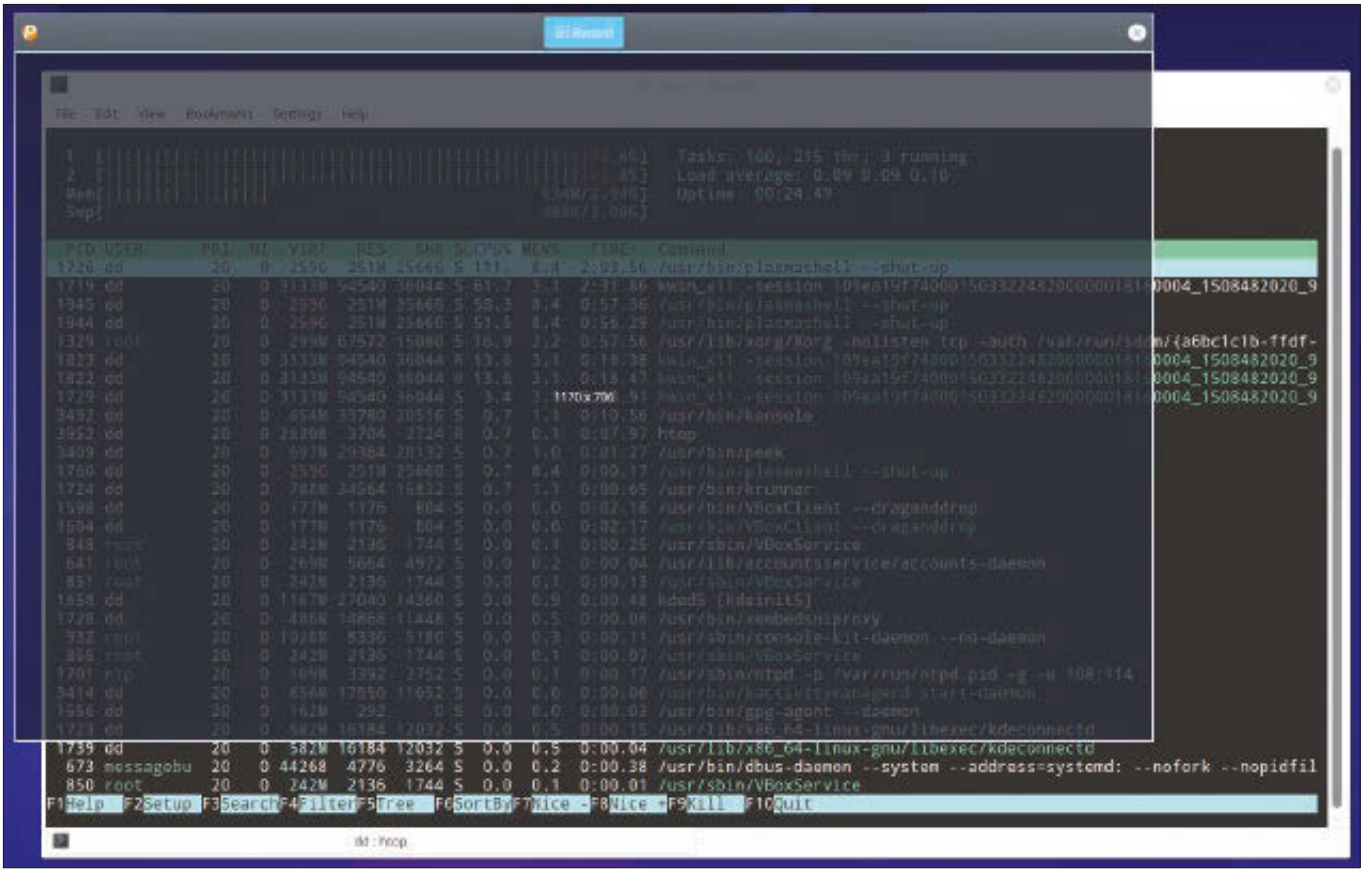
Pierwsze commity do Peeka pochodzą z grudnia 2015 roku. Od tamtej pory programiści aktywnie rozwijają program, a prace nabrały tempa na początku 2017 roku – codziennie zachodziły jakieś zmiany w kodzie. Jako względnie nowy program Peek nie trafił jeszcze do repozytoriów głównych dystrybucji.

W Ubuntu Peeka zainstalujemy jednak z repozytorium PPA (*Personal Package Archive*), natomiast w Arch Linuksie – z *Arch User Repository* (AUR) [5]. W przypadku innych dystrybucji – Fedory, Debiana czy Solusa – na stronie projektu znajdziemy instrukcje instalacji od programistów.

Jeśli z czymś Peek ma teraz trudności, to jest to przejście wielu dystrybucji na nowy serwer grafiki, Wayland. Z powodów bezpieczeństwa Wayland izoluje od siebie poszczególne pulpitowe aplikacje – jeden program nie może odczytywać zawartości okna innego programu. Z tego względu zrzuty ekranu całego pulpitu są teraz dużym wyzwaniem.

Ale nie jest to problem tylko Peeka – na aktualizacji Waylanda straciły również inne narzędzia do zrzutów ekranu, na przykład Shutter [6]. Ponadto Wayland nie podaje już bezwzględnych współrzędnych okna aplikacji, co przynajmniej teoretycznie ma otworzyć mu drogę na okrągłe lub zakrzywione ekrany 3D.

O ile nie korzystamy z Gnome i klasycznego serwera X pod menedżerem widoku *Gnome in Xorg*, o tyle Peek musi cofnąć się do warstwy kompatybilności Xwayland [7]. Ostatnio odbywa się to



Rysunek 3: Obszar do nagrywania wyznaczamy skalowalną ramką podającą swoje wymiary.

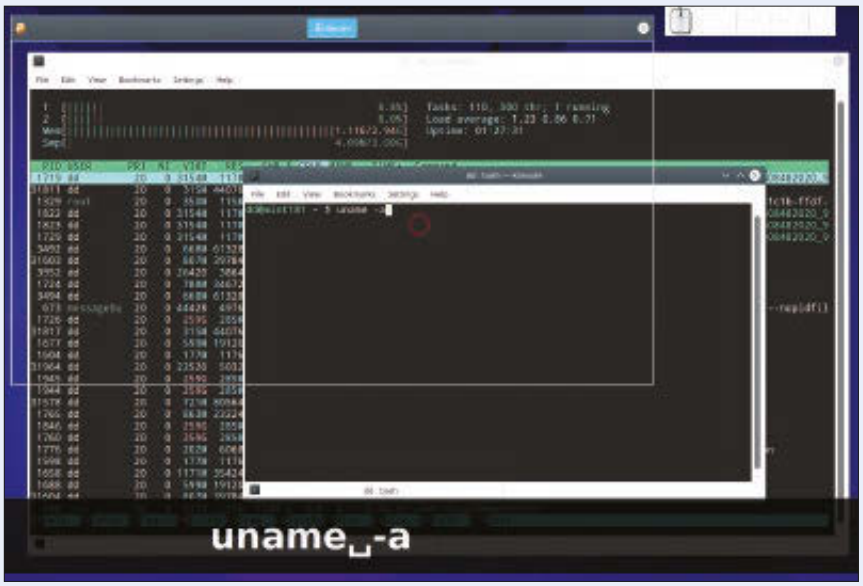
DEMONSTRACJA STEROWANIA

Jeśli zamiast zwykłego nagrania widoku chcemy pokazać widzom kolejne czynności, dobrze jest, aby na filmie widzieli wprowadzany tekst i ruchy myszą. W Linuksie mamy do tego dwie osobne aplikacje: key-mon [8] i screenkey [8]. Key-mon wyświetla okienko z symbolem myszy oraz klawiszami Ctrl, Alt i Shift. Screenkey pokaże natomiast na pasku wszystko, co wpisujemy z klawiatury.

Obie aplikacje w zależności od potrzeb mogą się nawzajem uzupełniać. Key-mon pokazuje ruchy myszą po wywołaniu w terminalu:

```
key-mon --visible_click
```

Wokół kursora i kliknięć program rysuje czerwone kółko. Screenkey pokaże natomiast, kiedy wciskamy Ctrl + C czy po prostu wyraźniej wyświetli wpisywane polecenia (Rysunek 4).



Rysunek 4: Key-mon i screenkey pokazują na wideo kliknięcia myszą i wciśnięcia klawiszy.

automatycznie po starcie programu, ale w razie potrzeby Peeka można uruchomić od razu pod Xwayland poleceniem:

```
GDK_BACKEND=x11 peek
```

Po spełnieniu tego warunku praca z programem staje się bardzo prosta: uruchomienie, ustawienie obszaru i nagrywanie. W ramce „Demonstracja sterowania” znajdziemy jeszcze garść porad o screencastach.

Podsumowanie

Lista ulepszeń Peeka i jednocześnie plan jego rozwoju na GitHubie [10] zawiera sporo pomysłów na przyszłość tego narzędzia. Najważniejszą sprawę już zaimplementowano – Peek obsługuje teraz prawdziwy format wideo (WebM), a w kolejce czekają inne przydatne funkcje: możliwość pauzowania nagrania, pływający wybór fragmentu ekranu i pasek postępu renderowania filmu.

INFO

[1] SimpleScreenRecorder: http://www.maartenbaert.be/simplescreenrecorder

[2] recordMyDesktop: http://recordmydesktop.sourceforge.net

[3] Peek: https://github.com/phw/peek

[4] LICEcap: http://www.cockos.com/licecap

[5] Peek w repozytorium AUR: https://aur.archlinux.org/packages/peek

[6] „Shutter nie działa z Waylandem”: https://bugs.launchpad.net/shutter/+bug/1502263

[7] XWayland: https://wayland.freedesktop.org/xserver.html

[8] key-mon: https://code.google.com/archive/p/key-mon

[9] screenkey: https://github.com/wavexx/screenkey

[10] Ulepszenia na przyszłość: https://github.com/phw/peek/issues?q=is%3Aissue+is%3Aopen+label%3Aenhancement

Warsztat admina: Inxi

Infotubisie

Nazwa narzędzia, któremu w tym miesiącu przyjrzy się Charly, może przypominać Teletubisie, jednak jego funkcje nie są bynajmniej dziecinne. Dzięki Inxi możemy uzyskać dokładne i szczegółowe informacje na temat sprzętu i oprogramowania zainstalowanego w danym systemie. Charly Kuehnast

Każdy administrator wie, jak uzyskać informacje na temat danego systemu. Ile rdzeni ma procesor? *cat /proc/cpuinfo*! Czy *eth3* to interfejs gigabitowy? *ip l sh*! Jednak zamiast wielu różnych narzędzi wystarczy nam jedno: Inxi [1].

Załóżmy, że chcemy uzyskać dane na temat maszyny, na której zazwyczaj nie pracujemy. Wywołujemy więc Inxi bez parametrów i uzyskujemy podstawowe informacje na temat sprzętu (procesor, taktowanie, rozmiar dysku) i oprogramowania (jądro i procesy). Jeśli chcemy sprawdzić dodatkowe szczegóły, parametr -F pozwoli nam zapoznać się z informacjami dotyczącymi sprzętu audio i wideo, partycji, RAID, temperatur i szybkości wentylatorów (Rysunek 1).

Jeśli interesuje nas tylko jeden komponent, możemy go sprawdzić odpowiednim parametrem, takim jak *-C*, *-A* czy *-G*, które służą do wyświetlania informacji na temat – odpowiednio – procesora, dźwięku i grafiki. Z kolei pamięć sprawdzimy opcją *-m* (małą literą – trzeba się do tego przyzwyczaić).

```
charly@funghi:~$ inxi -F -c4 -x
System:    Host: funghi Kernel: 4.4.0-78-generic x86_64 (64 bit gcc: 5.4.0) Console: tty 0
           Distro: Ubuntu 16.04 xenial
Machine:   Mobo: ASUSTeK model: P8Z68-V PRO v: Rev 1.xx Bios: American Megatrends v: 3603 date: 11/09/2012
CPU:       Quad core Intel Core i7-2600K (-HT-MCP-) cache: 8192 KB
           flags: (lm nx sse sse2 sse3 sse4_1 sse4_2 ssse3 vmx) bmips: 28020
           clock speeds: max: 5900 MHz 1: 1629 MHz 2: 1599 MHz 3: 1607 MHz 4: 1608 MHz 5: 1606 MHz
           6: 1697 MHz 7: 1602 MHz 8: 1689 MHz
Graphics:  Card: NVIDIA Device 1c03 bus-ID: 01:00.0
           Display Server: X.org 1.18.4 drivers: nvidia (unloaded: fbdev,vesa,nouveau)
           tty size: N/A Advanced Data: N/A out of X
Audio:     Card-1 NVIDIA Device 10f1 driver: snd_hda_intel bus-ID: 01:00.1 Sound: ALSA v: k4.4.0-78-generic
           Card-2 Intel 6 Series/C200 Series Family High Definition Audio Controller
           driver: snd_hda_intel bus-ID: 00:1b.0
Network:   Card: Intel 82579V Gigabit Network Connection
           driver: e1000e v: 3.2.6-k port: f040 bus-ID: 00:19.0
           IF: eth0 state: up speed: 1000 Mbps duplex: full mac: 14:da:e9:4f:8c:cb
Drives:    HDD Total Size: 512.1GB (16.5% used) ID-1: /dev/sda model: TS256GSSD340 size: 256.1GB
           ID-2: /dev/sdb model: TS256GSSD340 size: 256.1GB
Partition: ID-1: / size: 227G used: 72G (34%) fs: ext4 dev: /dev/sdb1
           ID-2: swap-1 size: 8.55GB used: 0.00GB (0%) fs: swap dev: /dev/sdb5
RAID:      No RAID devices: /proc/mdstat, md_mod kernel module present
Sensors:   System Temperatures: cpu: 29.8C mobo: 27.8C gpu: 0.0:
           Fan Speeds (in rpm): cpu: 0
Info:      Processes: 195 Uptime: 17:24 Memory: 586.5/7950.4MB Init: systemd runlevel: 5 Gcc sys: 5.4.0
           Client: Shell (bash 4.3.481) inxi: 2.2.35
```

Rysunek 1: Inxi nie skąpi wiadomości na temat systemu.

```
charly@funghi:~$ inxi -C -c4
CPU:       Quad core Intel Core i7-2600K (-HT-MCP-) cache: 8192 KB
           clock speeds: max: 5900 MHz 1: 1603 MHz 2: 1679 MHz 3: 1600 MHz 4: 1609 MHz 5: 1620 MHz
           6: 1668 MHz 7: 1977 MHz 8: 1674 MHz
charly@funghi:~$ sudo inxi -m  -c4
Memory:    Array-1 capacity: 32 GB devices: 4 EC: None
           Device-1: ChannelA-DIMM0 size: 2 GB speed: 1600 MHz type: DDR3
           Device-2: ChannelA-DIMM1 size: 2 GB speed: 1600 MHz type: DDR3
           Device-3: ChannelB-DIMM0 size: 2 GB speed: 1600 MHz type: DDR3
           Device-4: ChannelB-DIMM1 size: 2 GB speed: 1600 MHz type: DDR3
charly@funghi:~$
```

Rysunek 2: Jeśli uruchomimy Inxi z uprawnieniami administratora, program nagrodzi nas szczegółowymi informacjami na temat banków pamięci w komputerze.

O pamięci

Po uruchomieniu z uprawnieniami administratora Inxi opowie nam więcej na temat RAM-u: okaże się, że w maszynie testowej zainstalowano 4 2-gigabajtowe kości DDR taktowane 1600 MHz, co jest dość niską wartością jak na współczesne standardy (Rysunek 2). Widoczny na rysunkach 1 i 2 parametr *-c4* odpowiada za motyw graficzny. Domyślny motyw jest mało czytelny na terminalach z jasnym tłem, możemy to jednak z łatwością zmienić, korzystając z jednej z opcji od *-c1* do *-c32*.

Możemy nawet zaprząc Inxi do prostego monitorowania systemu. Jeżeli chcemy sprawdzić pięć (to domyślna liczba) procesów, które zużywają obecnie najwięcej RAM-u, użyjemy opcji *inxi -t m*. Jeśli interesuje nas dziesięciu głównych winowajców, napiszemy *-t m10*. Jeżeli natomiast słyszymy buczenie wiatraczka, zamiast *m* umieszczamy *c*, by sprawdzić obciążenie procesora. Możemy też połączyć obie opcje, by sprawdzić, które procesy zużywają najwięcej RAM-u i najbardziej obciążają procesor.

I wisienka na torcie: po przeglądzie sprzętu i oprogramowania możemy rzucić okiem na pogodę: *inxi -w Warsaw, Poland* powie nam, jak wygląda sytuacja na zewnątrz serwerowni. ■■■

CHARLY KÜHNAST

Charly Kühnast jest administratorem systemów operacyjnych w centrum danych w Moers w Niemczech. Do jego zadań należy zapewnienie bezpieczeństwa i dostępności danych. W wolnym czasie zajmuje się gotowaniem, rybkami akwariowymi i nauką japońskiego.

INFO

[1] Inxi: https://github.com/smxi/inxi

Odtwarzanie i edycja dźwięku i wideo w wierszu poleceń

Melt

Bez względu na to, czy jesteśmy początkujący, czy też nie, z Meltem nauczymy się edytować audio i wideo w wierszu poleceń. Bruce Byfield

Kiedy użytkownik wolnego oprogramowania myśli o nazwie Melt, zwykle przychodzi mu do głowy GCC MELT [1], popularne rozszerzenie do kompilatora GCC. Jednak odtwarzacz multimediów Melt [2] jest równie interesujący, gdyż wspiera wszystkie formaty plików, jakie znamy, a nawet klika, o których nie słyszeliśmy. Przyznać trzeba, że jego niestandardowa składnia wymaga trochę nauki, ale przecież złożoność poleceń zależy jedynie od naszych wymagań.

Melt to część pakietu MLT (Media Lovin' Toolkit) [3], wieloplatformowego zestawu narzędzi zaprojektowanego dla telewizji. Dwie kwestie sprawiają, że zwraca na siebie uwagę: mała ilość zależności i działanie w ramach zainstalowanych bibliotek i aplikacji. Jest to możliwe dzięki jego modularnej budowie i wysokopoziomowym powiązaniom z najważniejszymi językami programowania, takimi jak C++, Java, Lua, Perl, PHP, Python, Ruby i Tcl. Dodatkowo Melt to program nowoczesny, korzystający z możliwości wielu rdzeni oraz procesorów graficznych.

Pod względem funkcjonalnym jest to w pełnoprawny edytor, za pomocą którego możemy zmieniać jednorazowo lub na stałe zarówno dźwięk, jak i obraz. Prawdę mówiąc, początkowo był tylko programem testowym w MLT, jednak dzięki swojej wszechstronności wkroczył na wyższy poziom, stał się narzędziem wiersza poleceń, z możliwościami dostępnymi zwykle tylko aplikacjom pulpitu. Dziś często znajdziemy go w repozytoriach większych dystrybucji – jest niezależny od MLT.

BRUCE BYFIELD

Bruce Byfield jest dziennikarzem IT, autorem artykułów i redaktorem specjalizującym się w FOSS. Oprócz pisania prowadzi kursy komputerowe, zarówno na żywo, jak i poprzez platformy e-learningowe. W wolnych chwilach pisze o sztuce Wybrzeża Północno-Zachodniego. Więcej na temat jego pracy znajduje się na stronie http://brucebyfield.wordpress.com.

Odtwarzanie

Jak w przypadku większości poleceń, podstawowe działanie Melta ma prostą składnię. Aby uruchomić strumień audio lub wideo, wpisujemy:

```
melt PLIK
```

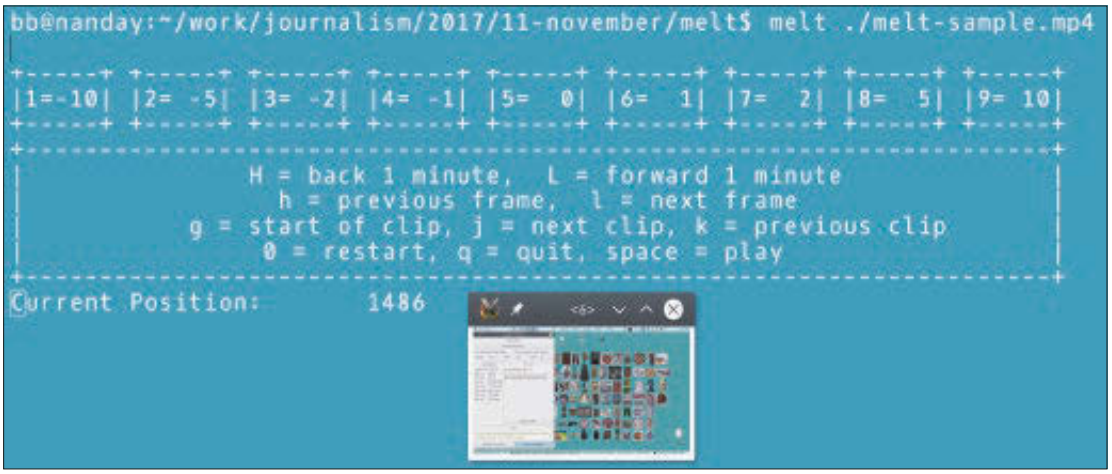
Pojawi się wtedy nowe okno z wideo. Jeśli jest to plik dźwiękowy, będzie ono puste. W przypadku wideo bez dźwięku przywita nas sam obraz. Kiedy odtworzone zostaną wszystkie pliki, okno musimy ręcznie zamknąć (Rysunek 1).

Na wejściu możemy podać większą liczbę plików, oddzielając je spacjami. Opcje dla każdego z nich podajemy po nazwie. Możemy też skorzystać z opcji *-group* specyfikowanej po podstawowym poleceniu, aby podane opcje obejmowały wszystkie pliki. Przykładowo:

```
melt -group in=0 out=25 PLIK1 PLIK2
```

odtworzy pierwszych 25 klatek z PLIK1, a następnie pierwszych 25 klatek z PLIK2. Oczywiście, jeśli chcemy odtworzyć konkretny fragment, zlokalizowanie numerów klatek wymaga kilku podejść.

Opcja grupowania odnosi się do plików wymienionych aż do pojawienia się kolejnego wystąpienia *-group*. Jeśli



Rysunek 1: Melt otwiera okno, w którym odtwarzane jest wideo.

chcemy odtworzyć pierwszych 25 klatek z PLIK1, a następnie cały PLIK2, polecenie wygląda następująco:

```
melt -group in=0 out=25 PLIK1
 -group PLIK2
```

W efekcie powstanie nowa, pusta grupa dla drugiego pliku. Możemy jednak dodać do niej inny zakres klatek do odtworzenia:

```
melt -group in=0 out=25 PLIK1
 -group in=35 out=60 PLIK2
```

Odtwarzając film, możemy skorzystać z opcji *-audio-track* lub *-hide-video*, aby wyłączyć wideo i przesłuchać jedynie ścieżkę dźwiękową. Analogicznie, opcje *-video-track* lub *-hide-audio* odtworzą jedynie wideo, bez dźwięku.
Poza *in=* i *out=* odtwarzanie można modyfikować na wiele innych sposobów. Jeśli próbujemy określić dokładne klatki do wyświetlenia, dodanie opcji *-progress* umożliwi przeglądanie graficznego odpowiednika odtwarzanego pliku. Dodatkowo możemy zapętlić dany utwór za pomocą *-repeat ILE_RAZY*.

Serwisy i filtry

Wymienione opcje powinny wystarczyć do prostego odtwarzania. Jednak *melt* ma też wiele zaawansowanych możliwości, które dokumentacja programu określa mianem „serwisów”.

Jeśli chcemy dodać serwis do wszystkich wymienionych plików, piszemy na końcu polecenia:

```
-group -filter SERWIS
```

Na przykład, aby odtworzyć w skali szarości pliki z aktualnego katalogu, piszemy:

```
melt -group in=0 out=25 PLIK1
 -group in=35 out=60 PLIK2
 -group -filter greyscale
```

Pamiętajmy o *-group*, gdyż w przeciwnym wypadku serwis zostanie zmodyfikowany opcjami wymienionymi przy poprzednim pliku.

Jednak jeśli chcemy, aby serwis był wykorzystany tylko w przypadku konkretnego strumienia, korzystamy z opcji:

```
-attach SERWIS:ARGUMENT
NAZWA=WARTOŚĆ
```

dodając ją po nazwie pliku tak jak każdą inną.

Melt posiada setki serwisów. Warto zapoznać się z ich listą, co zrobimy za pomocą opcji *-query*. Aby ograniczyć zwracane wiersze, możemy dodać argument:

```
melt -query TYP-SERWISU
```

W ten sposób zobaczymy grupę powiązanych serwisów (Rysunek 2). Nawet jednak wtedy warto dodać na końcu *|less*, aby całość była bardziej czytelna.

Zauważmy też, że wiele serwisów wymienionych jest z prefiksem, jednak wszystkie filtry podawane w poleceniu go nie mają. Sama lista niewiele nam powie, więc jeśli znajdziemy coś, co nas zainteresuje, warto skorzystać z opcji:

```
-query TYP-SERWISU=ID
```

dzięki której uzyskamy informacje na temat argumentów i wartości, jakie dany serwis przyjmuje. Jeśli takie informacje nie są dostępne, Melt wyświetli listę dostępnych serwisów z podziałem na ich typy.

Jeśli o typach mowa, to *-query* akceptuje następujące ich rodzaje:

- *consumers* (odbiorcy): aplikacje lub narzędzia odtwarzające plik, na przykład XML lub JACK. Domyślnie wykorzystywany jest SDL.
- *filters* (filtry): modyfikacje dźwięku lub obrazu. Nie zmieniają danego pliku, a jedynie sposób jego odtwarzania. Przykładowe filtry to zmiana nasycenia kolorów, głośności lub dodanie znaku wodnego.
- *producers* (producenci): biblioteki lub komponenty systemu, z których korzysta oprogramowanie odtwarzające.
- *transitions* (przejścia): sposób, w jaki odtwarzanie przechodzi z jednego pliku do drugiego. Na przykład *luma*, dzięki specjalnej bitmapie w skali szarości, zmienia jasność, a *matte* nakłada na siebie oba odtwarzane strumienie. Dodając do polecenia przejście, podajemy opcję *-mix DŁUGOŚĆ*, aby określić czas jego trwania, a zaraz potem nazwę: *-mixer PRZEJŚCIE*.
- *profiles* (profile): informacja o tym, jak należy przetworzyć plik (na przykład zmiana jego rozdzielczości). Jeśli nie podamy profilu, strumień nie będzie zmieniany. Początkujący mogą zignorować tę opcję.
- *presets* (ustawienia): opcje i format odtwarzania. Podobnie jak wcześniej, początkujący mogą to zignorować.
- *formats* (formaty): formaty audio, wideo oraz audio-wideo wspierane przez aplikację.

Możemy skorzystać z tej opcji, wymuszając dany format, jeśli Melt ma problemy z jego identyfikacją. Wiele z podanych formatów znanych jest jedynie ekspertom.

- *audio-codecs* (kodeki dźwięku): formaty audio wspierane przez aplikację. W połączeniu z *-query* zwróci krótszą listę niż słowo kluczowe *formats*.
- *video-codecs* (kodeki wideo): formaty wideo wspierane przez aplikację. Podobnie jak *audio-codecs*, w połączeniu z *-query*, zwróci krótszą listę niż *formats*.

Każdy z serwisów możemy uruchomić, podając jego typ, a następnie nazwę, na przykład *-TYP-SERWISU NAZWA*.

Aby uruchomić Melta, nie jest potrzebna dogłębna wiedza na temat jego składni. Jednak lista dostępnych serwisów zdradza, jak skomplikowane mogą być polecenia. Najprawdopodobniej przeciętny użytkownik nie korzysta z połowy dostępnych serwisów, ale jeśli ktoś jest ekspertem w dziedzinie audio-wideo, Melt równie dobrze spełni jego oczekiwania jak w przypadku początkującego.

Podsumowanie

Jak widać, polecenie *melt*, po dodaniu wszystkich potrzebnych nam opcji, może okazać się niezwykle skomplikowane. Powtarzanie ręcznego wprowadzania czegoś takiego jest kłopotliwe. Możemy co prawda skorzystać z historii wydawanych poleceń w Bashu, ale na dłuższą metę lepsze wydaje się zapisane polecenia:

```
-serialise
NAZWAPLIKU.melt
```

Aby je następnie odtworzyć, uruchamiamy je w aplikacji tak, jakby to był kolejny wspierany format. Dodamy do niego kolejny plik do odtwarzania z pomocą opcji *-track PLIK*.

```
bb@nanday:~$ melt -query audio_codecs
---
audio_codecs:
  - comfortnoise
  - s302m
  - aac
  - ac3
  - ac3_fixed
  - alac
  - dca
  - eac3
  - flac
  - g723_1
  - mlp
  - mp2
```

Rysunek 2: Początek listy formatów audio uzyskanej z pomocą opcji query wspieranych przez Melta.

Dociekliwi z pewnością będą chcieli dowiedzieć się więcej na temat formatu XML w MLT [4], dzięki któremu zapiszemy skomplikowane opcje odtwarzania, z wykorzystaniem tych samych komponentów co w poleceniu *melt*. Dokumentacja MLT dostępna w sieci zakłada zaawansowanie użytkownika w dziedzinie audio-wideo, co sprawia, że jest użyteczna jedynie we fragmentach, chyba że jesteśmy zdecydowani na wyszukiwanie definicji trudniejszych terminów lub na eksperymentowanie z oprogramowaniem.

Większości użytkowników podstawowa wiedza o Melcie wystarczy, aby z niego korzystać. Jak w przypadku większości tego typu programów, wliczając w to aplikacje pulpitu, Melt zatańczy tak, jak mu zagramy. Na stronach MLT znajdziemy informacje, że nie wszystko jeszcze działa poprawnie, na przykład znak wodny nie jest dodawany do wszystkich klatek. Jednak w większości przypadków Melt spełni oczekiwania nie tylko użytkowników pragnących jedynie odtworzyć dany plik, ale też tych, którzy potrzebują większych możliwości odtwarzania. ■■■

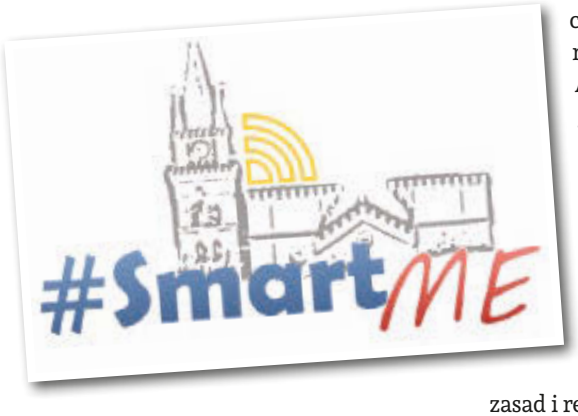
INFO

[1] GCC MELT: http://gcc-melt.org/
[2] Melt: https://www.mltframework.org/docs/melt/
[3] Media Lovin' Toolkit: https://www.mltframework.org
[4] MLT XML: https://www.mltframework.org/docs/mltxml/

Open source w praktyce

Zbiorowa inteligencja

Miasto Messyna we Włoszech staje się inteligentne dzięki open source i IoT. Swapnil Bhartiya



Wyobraźmy sobie, że jedziemy samochodem po mieście w środku nocy. Przed sobą widzimy rzędy czerwonych świateł. Ruchu prawie nie ma – aż chciałoby się przełączyć te wszystkie czerwone światła na zielone, by móc wcześnie dotrzeć do domu. A gdybyśmy tak mogli wyciągnąć telefon, automatycznie połączyć się z pobliskimi światłami i przełączyć je na zielone?

To właśnie spróbował zrobić zespół z Uniwersytetu w Messynie we Włoszech. „Udało się. Wszystko działało bardzo precyzyjnie w czasie rzeczywistym, istnieje jednak wiele reguł, zasad i regulacji, które trzeba przestrzegać, zanim w ogóle dany projekt zostanie wzięty pod uwagę" – powiedział Giovanni Merlino, badacz z Departamentu Inżynierii na Uniwersytecie w Messynie. Merlino wygłosił wykład na temat pracy swojego zespołu podczas konferencji Boston OpenStack Summit 2017.

Messyna to jedno z największych miast na Sycylii – mieszka w nim ponad 300 tysięcy osób. Tak duże miasto stoi przed różnymi wyzwaniami związanymi z zanieczyszczeniem środowiska, naprawą dróg, bezpieczeństwem obywateli, monitorowaniem i optymalizacją ruchu drogowego, dziurami w drogach itd.

Można podejść do problemu w nowoczesny sposób, budując infrastrukturę zbierającą dane za pośrednictwem rozmieszczonych po mieście urządzeń IoT. Dane te są następnie analizowane, by można było znaleźć rozwiązania różnych problemów. Weźmy np. pod uwagę dziury w jezdni. Jeśli czujniki zainstalowane w pojazdach bądź urządzeniach przenośnych wyposażonych w odpowiednią aplikację powiadomią władze o umiejscowieniu dziur, będzie je można naprawić. Jeśli miasto będzie monitorowało jakość powietrza, ciśnienie, jasność, wilgotność i inne czynniki środowiskowe, będzie można spróbować poprawić jakość życia w mieście.

SmartMe.IO

Rozwiązanie tych problemów leży zazwyczaj w rękach urzędników, zwykle jednak władze nie mają ani zasobów, ani motywacji, by inwestować w IT, by znaleźć odpowiednie rozwiązania. SmartMe.IO, akademicki zespół składający się z badaczy z Laboratorium Systemów Mobilnych i Rozproszonych (MDSLab) z Uniwersytetu w Messynie próbuje to zmienić, proponując niezależne od producenta, otwarte rozwiązania, które mogą pomóc miastom o niewielkich budżetach.

SmartMe.IO rozpoczął niewielki projekt crowdfundingowy związany z miastem [1], którego celem jest „zachęta to konwersacji z władzami Messyny, by zapoczątkować powstanie nowoczesnego ekosystemu wirtualnego opartego na paradygmacie Internetu rzeczy (IoT)".

SmartMe.IO zaczął od prototypów wykorzystujących sensory oparte na Arduino, by gromadzić dane związane ze środowiskiem,

Ilustracja główna © Andrey Kiselev, 123RF.com

takie jak wilgotność, jasność i poziom dwutlenku węgla. Nie poprzestano jednak na projekcie akademickim: celem był rozwój projektu w taki sposób, by mógł służyć ludziom poza Messyną. Zespołowi udało się przeprowadzić kampanię crowdfundingową rozwijającą ten projekt.

Krótko mówiąc, projekt SmartMe.IO składa się z trzech komponentów: (1) porozmieszczanych po mieście urządzeń IoT, które pobierają różne dane, (2) platformy IaaS, która zbiera i przetwarza te dane, oraz (3) przetwarzania danych, dzięki któremu można udostępnić miastu i jego mieszkańcom usługi najlepiej dostosowane do ich potrzeb.

Stack4Things

Zespół SmartMe.IO stworzył w pełni otwarty framework o nazwie Stack4Things: jest to kompletne rozwiązanie, które zdalnie zarządza flotą urządzeń IoT. Framework ten składa się z dwóch agentów: jeden jest po stronie serwera, a drugi – na urządzeniu końcowym. Agent po stronie serwera/chmury nosi nazwę IoTronic [2], natomiast ten po stronie węzła (urządzenia IoT) – Lightning-Rod. IoTronic jest usługą zarządzania zasobami dla OpenStacka, sprawdzonej platformy chmurowej o ogromnych możliwościach.

Twórcy SmartMe.IO chcieli zbudować oparte na IaaS rozwiązanie dla urządzeń IoT – platformę, którą da się skalować w zależności od potrzeb. Równocześnie zależało im na poszerzeniu zakresu wykorzystania OpenStacka poza tradycyjne ramy centrum danych, rozbudowując go o mechanizmy zarządzania zasobami związanymi z czujnikami i urządzeniami wykonawczymi. Na początku oprogramowanie zostało napisane w Node.js, później jednak przeniesiono je na Pythona w celu zapewnienia zgodności z OpenStackiem. Chmura działa obecnie w uniwersyteckim centrum danych.

Po stronie klienckiej Lightning-Rod działa na urządzeniach bazujących na jednopłytkowych komputerach, takich jak Raspberry Pi, na których działa zmodyfikowana wersja OpenWrt/LEDE wspólna dla wszystkich urządzeń obsługiwanych przez SmartMe.IO. Urządzenia te noszą nazwę Arancino i wyposażone są w bogaty zestaw czujników pobierających różnego rodzaju dane. Na uniwersytecie działało kiedyś laboratorium Arduino. Kiedy zostało zamknięte, eksperci z tego zespołu dołączyli do SmartMe.IO i rozpoczęli prace nad Arancino. W mieście znajduje się obecnie ponad 100 urządzeń IoT, które aktywnie gromadzą cenne dane.

Punkty wirtualne

Oprócz fizycznych urządzeń na mapie znajdują się punkty wirtualne, czyli obszary szczególnego zainteresowania – mogą to być albo określone lokalizacje, albo najczęściej klikane miejsca. Nie ma tam jednak zainstalowanych żadnych urządzeń.

„Wirtualne punkty są oznaczane w inny sposób, by poinformować użytkownika, że dane są pozyskiwane za pomocą ekstrapolacji czy prognozowania, a nie faktycznego próbkowania. Mechanizm prognozujący analizuje i koreluje serie danych z pobliskich urządzeń (np. czterech lub więcej sąsiadujących z danym punktem wirtualnym) i wprowadza oczekiwane wartości pomiarów homogenicznych (np. wilgotność z czterech pobliskich czujników wilgotności), a nawet heterogenicznych. System obsługuje więc popsute czujniki (niekompletne serie pomiarowe) i sprawdza się nawet na obszarach, na których zainstalowano niewiele czujników" – powiedział Merlino.

CKAN

Trzecim i najważniejszym elementem równania są dane. Platforma SmartMe.IO wykorzystuje otwarte oprogramowanie do zarządzania danymi, CKAN [3], do przetwarzania zgromadzonych danych, które są dostępne dla programistów w surowym formacie i można je swobodnie pobrać i korzystać z nich. Prawdziwą wartością dodaną ze strony SmartMe.IO jest ekstrapolacja tych danych i udostępnienie pomiarów w formie wygodnej do użycia, dzięki czemu władze miasta mogą je skorelować, np. po to, by zrozumieć główne przyczyny zanieczyszczenia powietrza i móc podjąć odpowiednie działania.

Twórcy SmartMe.IO nie zatrzymali się jednak na uruchomieniu i zarządzaniu urządzeniami IoT. Chcą, by te urządzenia stały się bardziej inteligentne i by w ten sposób oferowały większą wartość. Zaczęto np. wykorzystywać kamery, by móc sprawdzać liczbę osób gromadzących się w ruchliwych miejscach. Niektóre projekty działają już w miejscach takich jak lotniska, pomagając w zapewnieniu bezpieczeństwa. Kamery niosą jednak ze sobą inne ryzyko: zagrożenie dla prywatności poszczególnych osób. Aby je zminimalizować, SmartMe.IO implementuje na urządzeniach mechanizmy uczenia maszynowego w taki sposób, by żadne obrazy nie zostały zachowane ani przesłane na serwer. Przetwarzanie danych zachodzi na danym urządzeniu, serwer otrzymuje zaś wyłącznie dane statystyczne. Jak wspomnieliśmy, oprogramowanie jest całkowicie otwarte.

Przyszłość

Kolejnym krokiem będzie stworzenie pełnych rozwiązań, Arancino do chmury, dzięki czemu zainteresowane organizacje nie będą musiały się martwić o infrastrukturę i będą mogły korzystać z potrzebnych im danych.

Zespół SmartMe.IO pracuje też nad wykorzystaniem urządzeń przenośnych, takich jak smartfony. Aby projekt mógł się utrzymać, SmartMe.IO planuje sprzedaż usług, doradztwa i systemów zintegrowanych. Nie sprzedaje natomiast danych, które cały czas są otwarte.

Platforma SmartMe.IO jest częścią społeczności Fiware [4], europejskiej inicjatywy mającej na celu „zbudowanie otwartego i samoutrzymującego się ekosystemu skupionego wokół wolnych od opłat i sterowanych implementacjami standardów oprogramowania, ułatwiających rozwój nowych Aplikacji Inteligentnych w wielu sektorach".

SmartMe.IO jest znakomitym przykładem na to, że technologie open source umożliwiają każdemu stworzenie innowacyjnych rozwiązań realnych problemów. W miarę rozwoju projektu zwiększają się możliwości jego zastosowań, np. pomoc miastom w walce z zanieczyszczeniami, zarządzanie infrastrukturą, kontrola stanu dróg publicznych, ochrona przeciwpożarowa, kontrola ruchu drogowego, bezpieczeństwo w miejscach publicznych i wiele innych. ■■■

INFO

[1] SmartME project: http://smartme.unime.it
[2] IoTronic: https://github.com/openstack/iotronic
[3] CKAN: https://ckan.org/
[4] Fiware: https://www.fiware.org/devguides/hosting-your-application-on-a-fiware-cloud/

Budujemy radio FM za pomocą RTL-SDR

Radio na Raspberry Pi

RTL-SDR to tanie urządzenia USB, które potrafią odczytywać fale radiowe o częstotliwościach od 24 do 1766 MHz. Mają wiele interesujących zastosowań; jednym z bardziej praktycznych jest zbudowanie prostego radia FM. Pete Metcalfe

Istnieje wiele możliwości zbudowania radia FM na bazie Raspberry Pi, np. za pomocą układów RDA5807M czy TEA5767. Rozwiązania te działają, mają jednak swoje wady. Układ RDA5807M ma ubogą dokumentację i biblioteki tylko dla Arduino, TEA5767 zaś nie obsługuje kontroli głośności.

Tunery radiowe, takie jak RDA5807M i TEA5767, wykorzystują niskopoziomowy interfejs I2C. Natomiast radia definiowane programowo (SDR) oferują interfejs wyższego poziomu, który daje nam dostęp do mikserów, filtrów, wzmacniaczy, modulatorów i demodulatorów oraz detektorów sprzętowych.

Rysunek 1: Konfiguracja sprzętowa radia FM.

Istnieje wiele układów obsługujących SDR; najpopularniejsze są tanie klucze USB RTL-SDR [1]; można je kupić za 40–70 zł. SDR-y używane są do wielu rzeczy: śledzenia samolotów, monitorowania danych udostępnianych przez satelity okołoziemskie, odbierania sygnału telewizyjnego itp.

Zaczynamy

Aby zainstalować oprogramowanie, wydajemy polecenie:

```
sudo apt-get install rtl-sdr
```

Użyliśmy Raspberry Pi 1, ale całą procedurę przetestowaliśmy też na wersach 2 i 3 oraz na starym pececie z Lubuntu.

Ważna różnica między RTL-SDR a modułem tunera FM polega na tym, że do modułu możemy bezpośrednio podłączyć głośniki czy słuchawki, natomiast jeśli używamy RL-SDR, dźwięk generowany jest przez Raspberry Pi (lub PC). W naszych testach z Raspberry Pi użyliśmy głośników z zewnętrznym zasilaniem (Rysunek 1), w testach z laptopem zaś – wbudowanych głośników.

Klucz USB RTL-SDR zawiera także antenę zewnętrzną; jeśli to możliwe, powinniśmy ją umieścić blisko okna.

Działające w wierszu poleceń narzędzie *rtl_fm* to demodulator FM, który odczytuje i próbkuje wybraną

częstotliwość. Jeśli jesteśmy zainteresowani bardziej poważnymi zastosowaniami, powinniśmy się przyjrzeć projektowi GNU Radio [2].

Istnieje wiele opcji, które można przekazać rtl_fm; kluczowe to częstotliwość (*-f*), częstotliwość próbkowania (*-s*) oraz częstotliwość wyjściowa (*-r*). Wyjście *rtl_fm* powinno zostać przekierowane do programu odtwarzającego dźwięk. Użyliśmy *aplay*, działającego w wierszu poleceń odtwarzacza ALSA, lecz równie dobrze moglibyśmy użyć innego podobnego narzędzia. Jeśli chodzi o częstotliwość próbkowania (*-r*) *aplay*, dopasowaliśmy wyjściową częstotliwość próbkowania *rtl_fm*. Jako format próbek (*-f*) użyliśmy 16-bitowego LE (S16_LE).

Składnia wymagana do odtwarzania dźwięku nadawanego przez stację FM na częstotliwości 107,9 MHz to:

```
rtl_fm -f 107.9e6 -s 200000 -r 48000 |
 aplay -r 48000 -f S16_LE
```

Aplikację *rtl_fm* musimy zatrzymać, kiedy chcemy odtwarzać nową stację radiową. Jeśli program działa w tle, identyfikator procesu można znaleźć za pomocą *ps*, a następnie zakończyć proces poleceniem *kill*:

```
pi@raspberrypi:~ $ ps -e | grep rtl_fm
 1709 pts/0    00:00:33 rtl_fm
pi@raspberrypi:~ $ kill 1709
pi@raspberrypi:~ $ ps -e | grep rtl_fm
pi@raspberrypi:~ $
```

Regulacja głośności

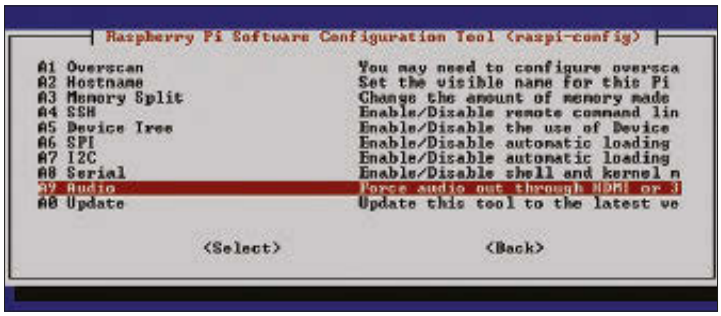
Raspberry Pi umożliwia wymuszenie przesyłania dźwięku przez HDMI lun gniazdo jack za pomocą *raspi-config* (menu *Advanced* | *Audio*) (Rysunek 2).

Istnieje kilka sposobów regulacji głośności. Bodaj najprostszy to użycie narzędzia *amixer*. Aby zmienić głośność dźwięk wysyłanego na głośniki zewnętrzne Raspberry Pi (lub wewnętrzne głośniki laptopa), używamy urządzenia PCM. Jeśli np. chcemy ustawić głośność na 70%, użyjemy poniższego polecenia:

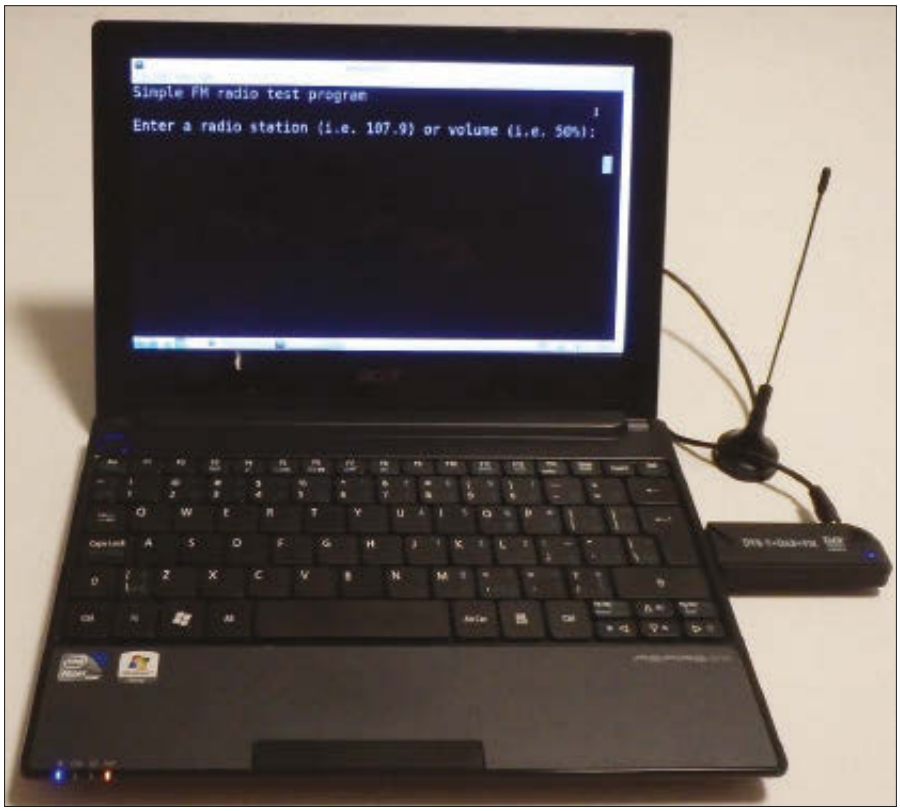
```
amixer sset "PCM" 70%
```

Testowy program w Pythonie

Do celów testowych stworzyliśmy prostą pythonową aplikację działającą w wierszu poleceń, którą można



Rysunek 2: Opcja *Advanced* | *Audio* narzędzia *raspi-config*.



Rysunek 3: Klucz RTL-SDR na laptopie z Lubuntu.

Rysunek 4: Radio FM z interfejsem LCD.

Listing 1: Radio działające w wierszu poleceń

```
# FM_radio.py
# Proste narzędzie testujące radio FM za pomocą klucza USB RTL SDR

import subprocess, signal, os, time

def newstation(station):
   global process, stnum
   # tworzymy wiersz poleceń rtl_fm i wprowadzamy nową częstotliwość
   part1 = "rtl_fm -f "
   part2 = "e6  -s 200000 -r 48000 - | aplay -r 48000 -f S16_LE"
   cmd = part1 + station + part2

   print ('Odtwarzanie stacji :', station)

   # Kończymy starsze połączenie FM, jeśli nadal działa
   if process != 0:
      process = int(subprocess.check_output(["pidof","rtl_fm"] ))
      print ("Process pid = ", process)
      if process != 0:
         os.kill(process,signal.SIGINT)
         time.sleep(2) # czekamy 2 sekundy i restartujemy rtl_fm

   # uruchamiamy nowe połączenie
   print (cmd)
   process = subprocess.Popen(cmd, shell=True)


def setvolume(thevolume):
   # przekazujemy nowe ustawienia głośności poleceniu amixer
   os.system('amixer sset "PCM" ' + thevolume)
   print ('głośność = ' , thevolume)

process = 0

print ("Prosty program testujący radio FM \n")
while True:
   answer = raw_input("Wprowadź stację radiową (np. 107.9) lub głośność (np. 50%): ")

   if answer.find('%') > 0:
      setvolume(answer)
   else:
      newstation(answer)
```

uruchomić zarówno Raspberry Pi, jak i na starym laptopie z Lubuntu (Rysunek 3).

Użyliśmy kilku bibliotek pythonowych. Za pomocą modułu *subprocess* uruchomiliśmy *rtl_fm* i pozyskaliśmy identyfikator procesu, natomiast biblioteka *os* została wykorzystana do zakończenia procesu. Dzięki modułowi *time* dodaliśmy opóźnienie, dzięki czemu *rtl_fm* zyskał wystarczająco dużo czasu na czyste zakończenie pracy przed ponownym uruchomieniem.

Funkcja *newstation* została utworzona po to, by zatrzymać działającą stację radiową, a następnie ponownie uruchomić *rtl_fm* z częstotliwością nowej stacji. Z kolei funkcja *setvolume* przekazuje nowe ustawienia głośności programowi *amixer*.

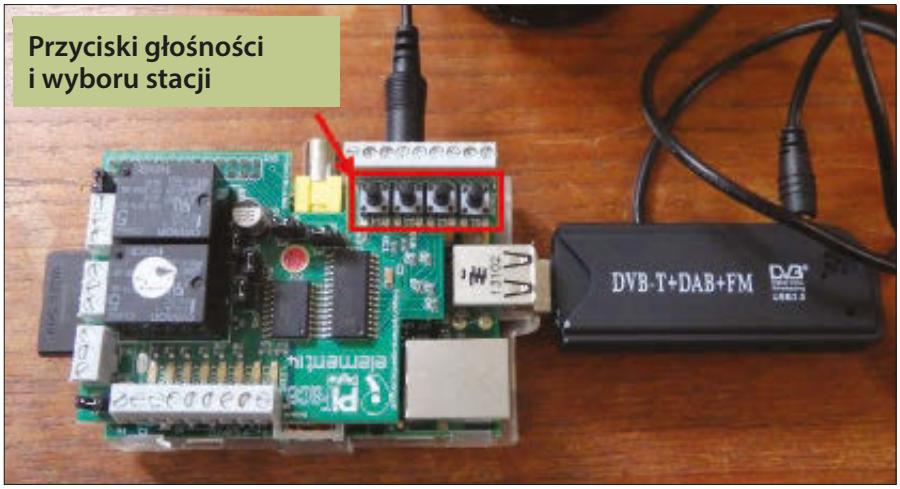
Program testowy (Listing 1) [3] przyjmuje poziom głośności w procentach lub częstotliwość. Możemy zatem wprowadzić albo stację radiową (np. 107.9), albo głośność (np. 50%).

Radio FM na Raspberry Pi

Podczas naszych testów próbowaliśmy wielu konfiguracji ułatwiających sprawowanie kontroli nad radiem. Ostatecznie okazało się, że shield LCD z przyciskami (Rysunek 4) sprawdza się najlepiej. Możemy też jednak użyć alternatywnych rozwiązań, takich jak PiFace (Rysunek 5).

Biblioteki LCD dla Pythona różnią się w zależności od użytego sprzętu, jednak większość shieldów LCD bazuje na bibliotekach Adafruit [4].

Kod Pythona dla shielda LCD (Listing 2) bazuje na prostym kodzie testowym z dodatkową logiką dla przycisków i kilku zdefiniowanych częstotliwości (zmienne *stations* i *sinfo*: powinniśmy wstawić odpowiednie wartości odpowiadające stacjom, których chcemy słuchać). Główna pętla sprawdza, czy przyciski nie zostały naciśnięte, używając funkcji *lcd.is_pressed*. Przyciski *LCD.UP* i *LCD.DOWN* służą do kontroli głośności, natomiast *LCD.LEFT* i *LCD.RIGHT* powodują przełączanie między zdefiniowanymi stacjami radiowymi. Funkcje *lcd.clear* i *lcd.message* są używane do wyświetlania informacji dotyczących nowych stacji radiowych na ekranie LCD. W celach testowych dodaliśmy 0,25-sekundowe opóźnienie do



Rysunek 5: Radio FM z interfejsem PiFace.

Listing 2: Radio FM na Raspberry Pi ze shieldem LCD

```
#!/usr/bin/python
# PI_FM_radion.py - Radio FM na Raspberry Pi ze shieldem LCD do regulacji głośności i wyboru stacji


import os, subprocess, signal
import time
import Adafruit_CharLCD as LCD

def newstation(direction):
    global stnum, stations, process

    print 'stnum=',stnum,'direction=',direction

    part1 = "rtl_fm -f "
    part2 = " -s 200000 -r 48000 | aplay -r 48000 -f S16_LE"

    if (stnum + direction < (len(stations)  )) and (stnum + direction > -1):
        stnum = stnum + direction
        print('Odtwarzanie stacji:', stations[stnum])
        cmd = part1 + stations[stnum] + part2
        if process != 0:
            process = int(subprocess.check_output(["pidof","rtl_fm"] ))
            print "Pid procesu = ", process
            os.kill(process,signal.SIGINT)
        # nawiązanie nowego połączenia FM
        print cmd
        process = subprocess.Popen(cmd, shell=True)


def setvolume(voldif):
    global thevolume
    if (thevolume + voldif > 0) and (thevolume + voldif <100):
        thevolume = thevolume + voldif
        os.system('amixer sset "PCM" ' + str(thevolume) + '%')
        print 'głośność = ' , thevolume


lcd = LCD.Adafruit_CharLCDPlate()

# Dodajemy własne stacje
stations = ['95.3e6','94.7e6','102.9e6','107.9e6']
sinfo = ['95.3 country', '94.7 light','102.9 easy','Y108 Rock\nHamilton ']
thevolume = 40        #głośność początkowa

stnum = 1             #wybór stacji domyślnej
process = 0
newstation(0)
lcd.message(sinfo[stnum])
setvolume(thevolume)


print 'Wyjście: Ctrl-C'
while True:
    if lcd.is_pressed(LCD.UP):
        setvolume(5)
        time.sleep(0.25)
    if lcd.is_pressed(LCD.DOWN):
        setvolume(-5)
        time.sleep(0.25)
    if lcd.is_pressed(LCD.LEFT):
        newstation(-1)
```

wszystkich klawiszy – możemy odpowiednio zmodyfikować tę wartość w zależności od potrzeb.

Wnioski

Własny odbiornik radiowy to tylko jedno z wielu zastosowań radia definiowanego programowo, przy czym główną rolę odgrywa tu tanie urządzenie USB RTL-SDR [5]. Bazując na prostym skrypcie przedstawionym w artykule, możemy łatwo rozbudować przedstawione rozwiązanie, dodając np. interfejs obsługiwany przez przeglądarkę, z poziomu którego moglibyśmy sterować radiem z innych pomieszczeń. ■■■

Listing 2: Radio FM na Raspberry Pi ze shieldem LCD cd.

```
        lcd.clear()
        lcd.message(sinfo[stnum])
        time.sleep(0.25)
    if lcd.is_pressed(LCD.RIGHT):
        newstation(1)
        lcd.clear()
        lcd.message(sinfo[stnum])
        time.sleep(0.25)
```

INFO

[1] O RTL-SDR: http://www.rtl-sdr.com/about-rtl-sdr/

[2] GNU Radio: https://www.gnuradio.org/

[3] Listingi do artykułu: ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/206/

[4] Biblioteki Adafruit: https://github.com/adafruit/Adafruit_Python_CharLCD

[5] Więcej projektów autora: https://funprojects.blog

Volumio 2.0: webowy odtwarzacz dźwięku

Odtwarzacz

Volumio i Raspberry Pi mogą nam pomóc rozbudować domowy system stereo o dodatkowe funkcje: obsługę wielu nowych formatów, strumieniowanie Spotify i odtwarzanie muzyki z zewnętrznych nośników i udziałów sieciowych. Pod wieloma względami zestaw Raspberry Pi z Volumio i oficjalnym wyświetlaczem przewyższa popularne rozwiązania komercyjne. Christoph Langner

Tradycyjne radio staje się coraz mniej popularne – jego rolę przejęły rozwiązania webowe umożliwiające strumieniowanie muzyki ze Spotify i od innych dostawców. Tego rodzaju urządzenia zaprojektowane przez gigantów przemysłu elektronicznego, takich jak Sony czy Panasonic, są jednak mało wygodne. Trudno znaleźć radio internetowe z dużym wyświetlaczem i wygodnym ekranem dotykowym. Poza tym zabawa zaczyna się dopiero wtedy, gdy możemy kontrolować radio aplikacją działająca na smartfonie czy tablecie. W tej kategorii ogromną popularnością cieszą się głośniki sieciowe (takie jak np. Sonos czy Raumfeld).

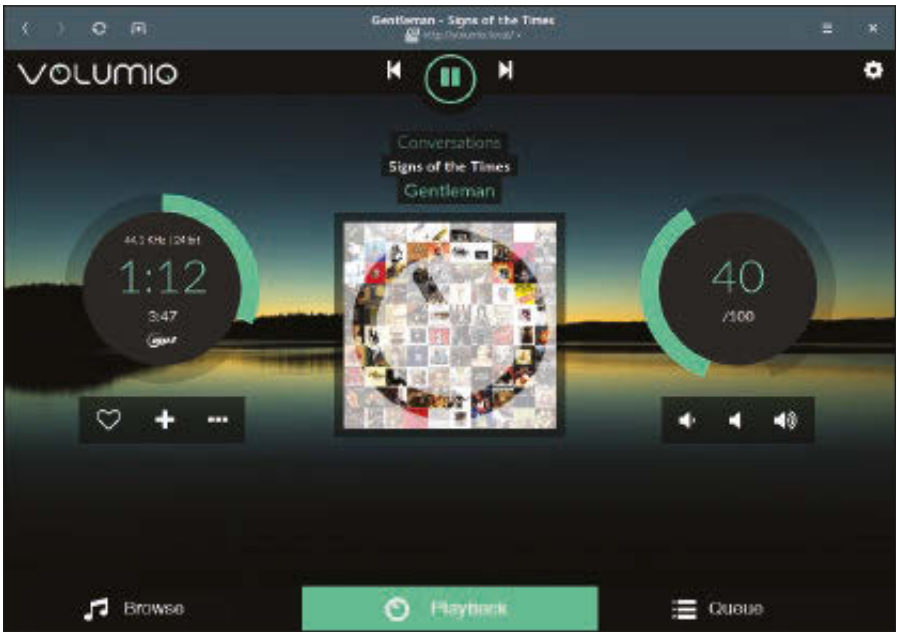
Tradycyjne systemy Hi-Fi – pozbawione wyświetlacza i połączenia sieciowego, ale z wyśmienitym brzmieniem, za które jeszcze kilka lat temu trzeba było słono płacić – nadal znajdziemy w wielu domach. Jeśli nie chcemy wymieniać wzmacniacza na najnowszy cud techniki, możemy łatwo rozbudować jego możliwości za pomocą dystrybucji Volumio [1]. Jest to oparte na Linuksie oprogramowanie do odtwarzania dźwięku cyfrowego, które łatwo skonfigurować i za pomocą którego przekształcimy dowolne radio czy zestaw stereo z wejściem line-in w „inteligentne” radio z dostępem sieciowym i strumieniowaniem Spotify. W połączeniu z Raspberry Pi i oryginalnym wyświetlaczem Raspberry Pi Display całość stanowi wyjątkowo łatwe w użyciu i eleganckie rozwiązanie.

Volumio 2.0

Od końca 2016 r. Volumio 2 dostępny był w znacznie ulepszonej wersji, która zawiera wiele innowacji i z której usunięto dużo błędów [2]. W porównaniu z poprzednimi wydaniami Volumio 2 został wzbogacony o mechanizm obsługi wtyczek, funkcję hotspot oraz ulepszony interfejs użytkownika.

Volumio 2 dla Raspberry Pi bazuje na bieżącej wersji Raspbiana Jessie. Oprogramowanie obsługuje też inne komputery jednopłytkowe, takie jak Odroid-C1/C2 czy CuBox-i.

Obraz dla klasycznych pecetów bazuje na Debianie. Aby móc korzystać



Rysunek 1: Interfejs WWW Volumio w przeglądarce internetowej.

z oprogramowania, zachowujemy obraz z Volumio na karcie SD, tak jak zrobilibyśmy z obrazem Raspbiana, a następnie uruchamiamy z niego malinkę. Na karcie powinniśmy mieć co najmniej 4 GB wolnego miejsca, jeśli natomiast chcemy przechowywać na niej część swojej kolekcji muzycznej, powinna mieć odpowiednio większy rozmiar.

Po instalacji i pierwszym uruchomieniu możemy uzyskać dostęp do Volumio z poziomu większości komputerów z Linuksem i macOS-em, używając URL-a *http://volumio.local*, lub bezpośrednio wpisując adres IP Raspberry Pi.

Jako że Volmio nie wyświetla adresu IP na ekranie podczas uruchamiania systemu, musimy go odczytać z w inny sposób. W Linuksie wystarczy użyć narzędzia *arp-scan*, które znajdziemy w repozytorium praktycznie każdej popularnej dystrybucji:

```
$ sudo arp-scan --localnet |
  grep Raspberry
192.168.111.195 b8:27:eb:66:ab:44
  Raspberry Pi Foundation
```

Jeśli podłączyliśmy urządzenie z Volumio do monitora, tuż po instalacji ujrzymy jedynie ekran logowania. W podstawowej konfiguracji Volumio jest skierowane do użytkowników, którzy chcą kontrolować odtwarzacz przez sieć, możemy jednak ławo skonfigurować interfejs graficzny.

Centrum dowodzenia

Volumio bazuje na Linuksie, mamy więc do dyspozycji serwer SSH z uwierzytelnianiem za pomocą kluczy publicznych, dzięki czemu możemy się zalogować do systemu poleceniem:

```
ssh volumio@volumio.local
```

lub podając odpowiedni adres IP zamiast *volumio.local*. Zarówno nazwa użytkownika, jak i hasło do *volumio*. Konto administratora jest wyłączone, jednak podobnie jak w innych dystrybucjach możemy wydawać polecenia z uprawnieniami administratora za pomocą *sudo*.

Volumio oferuje interfejs wiersza poleceń: wydając na maszynie z Volumio komendy takie jak *volumio pause* lub *volumio volume 50*, możemy kontrolować odtwarzacz bez przeglądarki czy klienta MPD. Przegląd wszystkich poleceń wyświetlimy za pomocą *volumio --help*.

GUI

Najważniejsze funkcje interfejsu webowego są oczywiste (Rysunek 1): u góry na środku znajduje się przycisk odtwarzania i pauzy, zaś przyciski po obu stronach pozwalają nam przechodzić do poprzedniego i następnego utworu na liście odtwarzania. Pod przyciskami znajdują się nazwy bieżącej ścieżki i albumu. Pokrętło po lewej stronie to miejsce odtwarzania bieżącej ścieżki, za pomocą którego możemy przewijać utwór, zaś prawe służy do kontroli głośności.

Aby zmienić język, otwieramy panel boczny, klikając ikonkę zębatki w prawym rogu, po czym przechodzimy do opcji *Appearance | Language* i wybieramy żądany język z listy rozwijanej *Select Language*. Po zachowaniu zmian przyciskiem *Save* interfejs zostanie przeładowany w nowym języku. W tym oknie możemy też wybrać obrazki tła i motywy graficzne interfejsu użytkownika.

Muzykę najprościej jest dodać, podłączając urządzenie USB z plikami MP3. Oprócz MP3 Volumio obsługuje formaty AAC, ALAC, FLAC, WAV i PLS. Program automatycznie montuje nośnik; swoją bibliotekę muzyczną możemy przeglądać z poziomu menu *Music Library | USB*. Inna możliwość to użycie Samby; w tym przypadku URL to *smb://volumio*.

Interfejs sieciowy Raspberry Pi ma kiepską przepustowość, ponieważ moduł sieci dzieli wewnętrzny interfejs USB z portami USB, więc przesłanie całej kolekcji muzycznej może długo trwać. Jeśli jesteśmy niecierpliwi, możemy odłączyć Raspberry Pi od zasilania, usunąć z urządzenia kartę SD i umieścić ją w czytniku kart podłączonym do komputera. Volumio szuka nowych utworów w *volumio_data* w katalogu */dyn/data/INTERNAL/*. Partycja sformatowana jest jako Ext4.

Wzmocnienie

Volumio domyślnie odtwarza dźwięk przez gniazdo słuchawkowe Raspberry Pi. Możemy je połączyć z domowym systemem stereo kablem minijack-RCA. Inna możliwość to przesyłanie audio kablem HDMI (menu *Playback*). Volumio obsługuje też wiele HAT-ów z przetwornikami DAC – są o płytki rozszerzające możliwości Raspberry Pi poprzez GPIO. Możemy dzięki nim zyskać znacznie wyższą jakość dźwięku i więcej wyjść audio (np. Toslink). Audiofile mogą nawet wyposażyć malinkę we wzmacniacz lampowy [4].

Jeśli pracujemy na Raspberry Pi 3 lub podłączyliśmy kompatybilną kartę Wi-Fi przez USB do starszej wersji urządzenia, Volumio automatycznie skonfiguruje hotspot Wi-Fi. SSID to *Volumio*, zaś hasło dostępu to *volumio2* (Rysunek 2). Dzięki temu podczas domowej imprezy możemy dać znajomym dostęp do swojej kolekcji muzycznej, nie musząc podawać im hasła do domowej sieci Wi-Fi (Rysunek 3).

Rysunek 2: Goście mogą zalogować się do maszyny z Volumio ze swoich telefonów komórkowych i przejąć rolę DJ-ów.

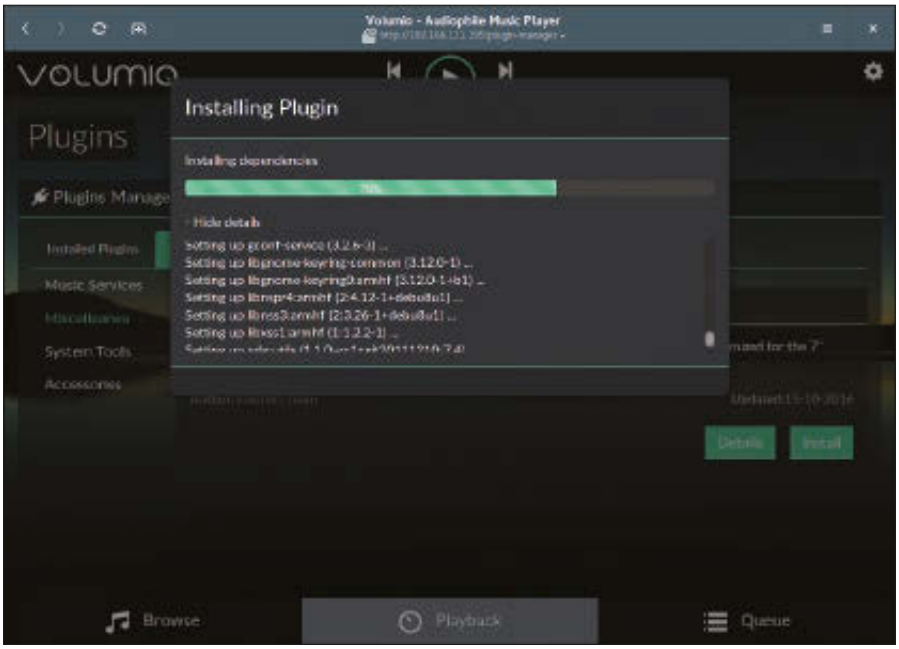
Rysunek 3: Interfejs webowy Volumio zmniejsza się, by dopasować się do wielkości ekranu smartfona.

Ustawienia hotspota możemy w razie potrzeby zmienić w menu *Network | Hotspot Settings*. Możemy też całkowicie wyłączyć dostęp przez Wi-Fi.

Wyświetlacz

Volumio został zaprojektowany z myślą o użytkownikach, którzy chcą podłączyć Raspberry Pi do domowego systemu stereo, ukrywając malinkę za wzmacniaczem. System można kontrolować albo za pomocą smartfona, albo przeglądarki na pececie. Jeśli dodamy oficjalny wyświetlacz i odpowiednią obudowę, ławo rozbudujemy domowe Wi-Fi o funkcje sieciowe, kontrolując go wygodnie za pomocą 7-calowego ekranu dotykowego.

W podstawowej konfiguracji monitor podłączony do Raspberry Pi z Volumio nie wyświetla niczego za wyjątkiem komunikatów startowych i ekranu logowania. System nie zawiera nawet graficznego środowiska użytkownika. Możemy jednak łatwo zainstalować dodatkowe funkcje (takie jak wyświetlanie interfejsu graficznego na podłączonym ekranie) za pomocą menu *Plugins* w panelu bocznym.



Rysunek 4: Dzięki wtyczkom możemy rozbudować Volumio o nowe funkcje, takie jak obsługa oficjalnego wyświetlacza Raspberry Pi.

Rysunek 5: Wtyczka Touch Display została zoptymalizowana pod kątem współpracy z oficjalnym wyświetlaczem Rapsbery Pi, obsługuje jednak również inne monitory.

Dostępne obecnie wtyczki należą do różnych kategorii; możemy je wyświetlić w zakładce *Search Plugins*. Wtyczkę *Touch Display Plugin* znajdziemy w *Miscelleanea*; moduł instalujemy w systemie, naciskając *Install*. Volumio zainstaluje wtedy serwer X i przeglądarkę Chromium, używając w tym celu systemowego menedżera pakietów Raspbiana, co może potrwać nawet kilkadziesiąt minut (Rysunek 4). Uwaga: przed instalacją należy odpiąć od urządzenia klawiaturę, w przeciwnym bowiem razie instalator się zawiesi.

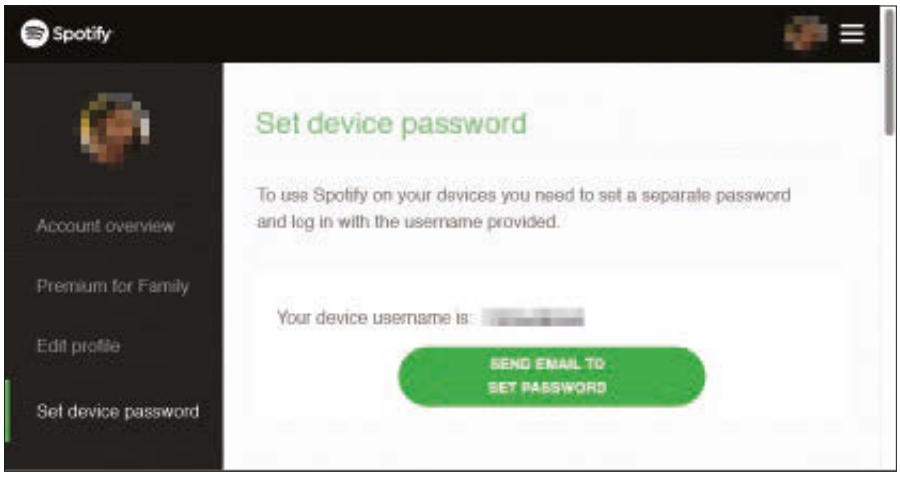
Na koniec musimy aktywować wtyczkę. W tym celu przechodzimy do zakładki *Installed Plugins* i przesuwamy kontroler *Touch Display* na *On*. Wkrótce potem powinniśmy ujrzeć środowisko graficzne ze stroną WWW Volumio. Ma ona minimalistyczny wygląd – została pozbawiona elementów, które rozpraszałyby nas przy słuchaniu; uniemożliwia też użycie Raspberry Pi do innych celów (Rysunek 5). Wtyczka konfiguruje system w taki sposób, że środowisko graficzne ładuje się automatycznie nawet po restarcie.

Spotify

Instalacja wtyczki Spotify z poziomu menedżera wtyczek wygląda podobnie jak włączanie obsługi ekranu dotykowego. Dzięki przeniesieniu obsługi Spotify do wtyczki programiści mogą szybko odpowiadać na zmiany związane z usługą, aktualizując samą wtyczkę.

Po instalacji włączamy wyczkę w zakładce *Installed Plugins*. Pojawi się wtedy przycisk *Settings*, po kliknięciu którego wprowadzamy dane specjalne dostępowe do Spotify, czyli specjalne hasło urządzenia, które znajdziemy na witrynie Spotify. Aby je uzyskać, logujemy się i przechodzimy do *Account overview* – w dolne części strony znajdziemy opcję *Set device password*, a w następnym oknie – nazwę użytkownika urządzenia. Jeśli teraz naciśniemy przycisk, Spotify wyśle nam mailem specjalne hasło (Rysunek 6). Dane te wprowadzamy w konfiguracji wtyczki Spotify na Volumio.

Następnie otwieramy bibliotekę z muzyką, klikając zakładkę *Browse* na głównym ekranie: powinniśmy ujrzeć tam również swoje listy dotwarzania Spotify. Volumio zainstaluje zarówno nasze własne listy odtwarzania, jak również inne oferowane przez usługę i nowe wydania;



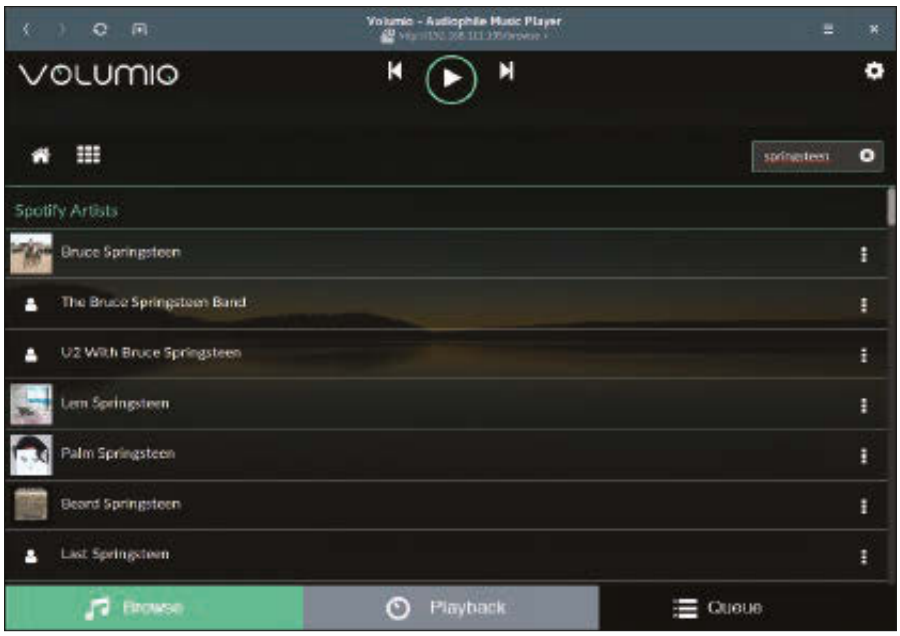
Rysunek 6: Potrzebne nam będzie hasło urządzenia z ustawień konta Spotify, by móc wygodnie korzystać ze Spotify na Volumio.

pozwala też filtrować muzykę według gatunku i nastroju. Kiedy przeglądamy kolekcję, promowane przez Spotify utwory znajdziemy w sekcji *Spotify Artists* (Rysunek 7). Rozważano również integrację z innymi usługodawcami, takimi jak Amazon Prime czy Google Play Music, na razie jednak Volumio ich nie obsługuje.

Architektura

Z technicznego punktu widzenia Volumio nie został zbudowany od zera, bazuje bowiem na serwerze dźwięku Music Player Daemon (MPD) [5], który rozwijany jest od lat i którego można kontrolować za pomocą różnych programów klienckich. Pełne wyjaśnienia dotyczące architektury znajdziemy na wiki projektu [6]. MPD znajdziemy w repozytoriach wielu dystrybucji Linuksa. Ulepszeniem ze strony Volumio jest dodanie interfejsu webowego i optymalizacja pod kątem interakcji z niewielkim komputerem, takim jak Raspberry Pi.

Volumio możemy też kontrolować za pomocą klasycznych klientów MPD [7], które są dostępne w wersjach na wiele systemów operacyjnych, w tym iOS-a i Androida [8] (Rysunek 8). Mają one jednak swoje wady. Choć Volumio umożliwia dostęp do montowanych nośników (np. przez USB czy przez sieć) z poziomu klientów MPD, na liście nie znajdziemy Spotify. Poza tym odtwarzanie utworów lokalnie koliduje ze Spotify, tj. podczas odtwarzania utworu na



Rysunek 7: Z poziomu Volumio możemy w łatwy sposób korzystać ze Spotify.

Spotify przez przeglądarkę możemy zacząć odtwarzać inny przez klienta MPD.

Aplikacja webowa Volumio, którą można znaleźć w sklepie Google Play Store, nie jest pozbawiona wad [9]. Choć może się połączyć do Raspberry Pi z Volumio, wyświetlacz pozostaje pusty. Problem ten widać w raportach o błędach zgłaszanych przez różnych użytkowników. Aby więc móc kontrolować Volumio za pomocą smartfona czy tabletu, najlepiej zignorować aplikacje i skorzystać z wersji przeglądarkowej.

Wiele pomieszczeń

Volumio ma jedną wadę w porównaniu z komercyjnymi systemami multiroom: odtwarzania nie można synchronizować między różnymi pomieszczeniami. Jest to jednak funkcja bardzo wyczekiwana przez fanów odtwarzania muzyki na Raspberry Pi. Jako że funkcja ta nie jest wbudowana w Volumio, postanowili ją stworzyć twórcy androidowej aplikacji Sound@home.

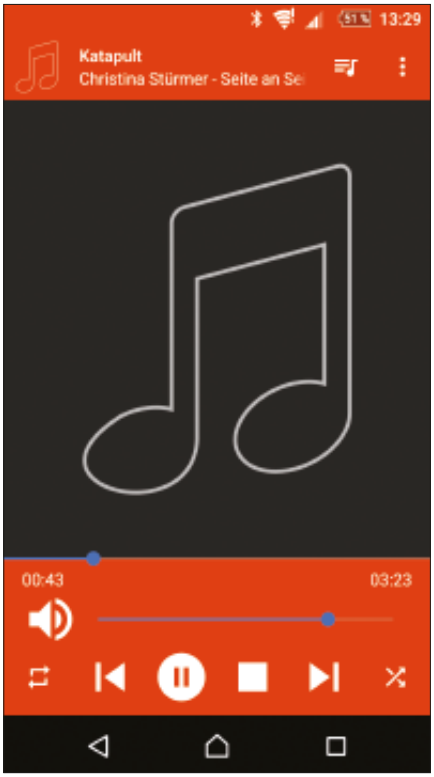
Problem w tym, że twórcom aplikacji trudno jest dotrzymać dynamicznego tempa rozwoju Volumio. Kiedy powstawał ten artykuł, Sound@home zoptymalizowany był pod kątem obsługi Volumio 1.55, choć aplikacja działała z bieżącą wersją serwera dźwięku, przy pewnych ograniczeniach. Wersja dla Volumio 2 pojawiła się w sierpniu 2017 roku [10]. Zawiera ona mechanizm automatycznej konfiguracji odsłuchiwania w wielu pomieszczeniach, obsługę wtyczki YouTube, ulepszenia związane z szybkością działania oraz funkcję długiego kliknięcia umożliwiającą ręczną konfigurację (Rysunek 9).

Wnioski

Od czasu ukazania się pierwszej wersji Volumio przeszedł wiele ulepszeń. Jeśli połączymy Raspberry Pi 3 z Volumio, wyświetlacz, obudowę i urządzenie wyjściowe, otrzymamy rozwiązanie, które znacznie przewyższa rozwiązania komercyjne. Volumio obsługuje się szybko i wygodnie – bez względu na to, czy używamy do tego gestów, przeglądarki WWW czy aplikacji działającej na smartfonie. Muzykę możemy odtwarzać z wielu źródeł, lokalnych i zdalnych, w różnych formatach; obsługiwana jest też popularna usługa Spotify.

Co istotne, Volumio ma aktywną społeczność programistów. Kod źródłowy interfejsu i głównych komponentów

możemy pobrać ze strony projektu na GitHubie [11]; możemy tam też zgłaszać błędy i własne sugestie dotyczące dalszego rozwoju projektu. Odpowiedzi na pytania dotyczące konfiguracji i kontroli Volumio znajdziemy na forum projektu[12]. ■■■



Rysunek 8: Możemy też kontrolować Volumio z poziomu innych klientów MPD, takich jak M.A.L.P. dla Androida, w ten sposób nie będziemy jednak mogli korzystać ze Spotify.

Rysunek 9: Androidowa aplikacja Sound@home pozwala korzystać z Volumio w wielu pomieszczeniach.

INFO

[1] Volumio: https://volumio.org

[2] Sklep z Volumio 2: https://volumio.org/volumio-2-stable-release/

[3] Fing: https://play.google.com/store/apps/details?id=com.overlook.android.fing

[4] Wzmacniacze lampowe dla Raspberry Pi: http://www.pi2design.com/502hta.html

[5] MPD: https://www.musicpd.org

[6] Prezentacja architektury Volumio: https://github.com/volumio/Volumio2/wiki

[7] Klienty MPD: https://www.musicpd.org/clients/

[8] M.A.L.P.: https://play.google.com/store/apps/details?id=org.gateshipone.malp

[9] Aplikacja webowa Volumio: https://play.google.com/store/apps/details?id=com.volumio.moritz.volumiowebapp_release

[10] Sound@home dla Volumio: https://play.google.com/store/apps/details?id=com.digx.soundhome

[11] Volumio na GitHubie: https://github.com/volumio

[12] Forum Volumio: https://volumio.org/forum

NAJNOWSZY NUMER

JEDYNY W POLSCE MAGAZYN O LINUKSIE

LINUX MAGAZINE

zawsze na allegro

bezpłatna dostawa

Instalujemy chmurę OpenStack za pomocą Packstacka

Z głową **w chmurach**

Instalacja OpenStacka na pierwszy rzut oka może wydawać się czarną magią, tymczasem w pełni funkcjonalną chmurę obliczeniową możemy uruchomić niewielkim nakładem pracy w stosunkowo krótkim czasie przy użyciu narzędzi automatyzacji. Grzegorz Juszczak

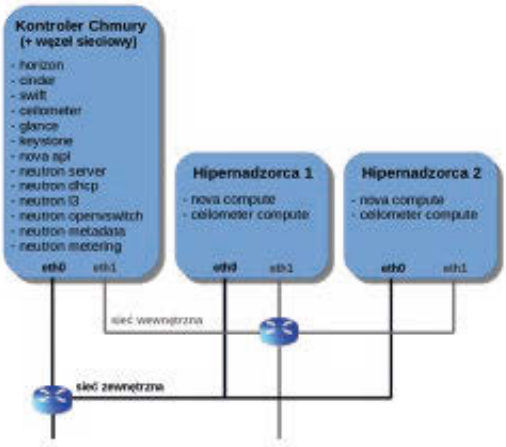
Od kilku lat na globalnym rynku IT, zarówno w infrastrukturze korporacyjnej, jak i wśród użytkowników prywatnych, coraz częściej pojawiają się chmury obliczeniowe. W średnich i dużych firmach zastępują one dotychczasową, niezbyt elastyczną, kiepsko skalowalną i nierzadko drogą infrastrukturę komputerową, a w przypadku małych firm, start-upów oraz użytkowników prywatnych dają możliwość przeniesienia całego ciężaru świadczenia usług IT na zewnętrznego dostawcę.

Poza podziałem na chmury prywatne i publiczne dzielimy chmury obliczeniowe również ze względu na rodzaje oferowanych usług. W tej kategorii prym wiodą głównie dwa rodzaje chmur: IaaS (Infrastructure as a Service), np. OpenStack, AWS (Amazon Web Services), dostarczające całą infrastrukturę wirtualną: serwery, pamięci masowe, sieć wirtualną oraz SaaS (Storage as a Service), np. Dropbox, Hightail, dostarczające swoim klientom przestrzeń dyskową.

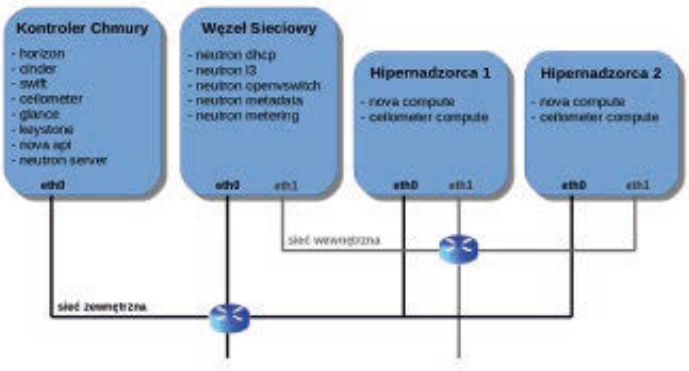
W niniejszym artykule przyjrzymy się bliżej środowisku OpenStack RDO – chmurze typu IaaS, rozwijanej przez agencję NASA oraz firmę Rackspace na licencji Open Source.

Istnieje kilka sposobów instalacji chmury OpenStack, począwszy od żmudnej instalacji poszczególnych serwisów OpenStacka krok po kroku, a skończywszy na zautomatyzowanych skryptach instalacyjnych, takich jak Packstack czy TripleO (OpenStack on OpenStack), znacznie przyspieszających instalacje chmury. O ile TripleO głównie używany jest w instalacjach środowisk produkcyjnych, o tyle Packstacka używa się do szybkiego stawiania środowisk deweloperskich lub demonstracyjno-poglądowych.

W niniejszym artykule przedstawimy przykładową instalację OpenStack RDO w wersji Pike przy użyciu instalatora Packstack.



Rysunek 1a: Przykładowe środowisko OpenStacka: Neutron zintegrowany z kontrolerem chmury.



Rysunek 1b: Przykładowe środowisko OpenStacka: usługi sieciowe (Neutron) na osobnym serwerze, zwanym węzłem sieciowym (ang. Network Node).

Komponenty OpenStacka

Chmura obliczeniowa jest pewnego rodzaju systemem usług rozproszonych pomiędzy węzłami (ang. nodes) wchodzącymi w jej skład. Do najważniejszych usług OpenStacka zaliczamy:

- Keystone: uwierzytelnianie i autoryzacja użytkowników
- Neutron: usługi sieciowe w chmurze
- Cinder: usługa dostarczania urządzeń blokowych dla instancji (maszyn wirtualnych)
- Nova: system zarządzania instancjami oparty na KVM (hipernadzorca)
- Glance: rejestr obrazów do tworzenia instancji
- Swift: przechowywanie plików w chmurze
- Ceilometer: silnik pomiarowy do zbierania danych i analiz płatniczych
- Heat: serwis do automatyzacji tworzenia kompleksowych projektów w chmurze
- Horizon: graficzny panel administracyjny do zarządzania chmurą z poziomu przeglądarki internetowej

Typowe środowisko OpenStacka składa się z kontrolera chmury (ang. controller node), na którym działa

większość usług w chmurze, w tym również usługi sieciowe (Neutron) oraz co najmniej jednego hipernadzorcy (ang. compute node lub hypervisor) kontrolującego pracę maszyn wirtualnych, zwanych instancjami (Rysunek 1a). Najczęstszym powodem integracji kontrolera chmury z węzłem sieciowym na jednym i tym samym serwerze jest nieduży rozmiar chmury oraz dostęp do stosunkowo wydajnego serwera, który jest w stanie obsłużyć odpowiednią liczbę komponentów OpenStacka.

W dużych środowiskach, gdzie mamy do czynienia z dużym ruchem sieciowym i dużą liczbą obsługiwanych hipernadzorców, najczęściej usługi sieciowe (Neutron) są przeniesione na osobny serwer – węzeł sieciowy (ang. network node) (Rysunek 1b).

Środowisko testowe

Do instalacji naszej chmury użyjemy trzech serwerów (ang. nodes) o następujących parametrach sprzętowych:

1. Kontroler chmury (ang. Controller):
   CPU: 2.5GHz (4 rdzenie)
   RAM: 16GB
   HDD: 160GB
   interfejsy: 2xGbE; eth0 (sieć dostępowa), eth1 (sieć wewnętrzna)
   system operacyjny: CentOS 7 (64bit)
2. Hipernadzorca 1 (ang. Compute 1):
   CPU: 2.5GHz (4 rdzenie)
   RAM: 16GB
   HDD: 160GB
   interfejsy: 2xGbE; eth0 (sieć dostępowa), eth1 (sieć wewnętrzna)
   system operacyjny: CentOS 7 (64bit)
3. Hipernadzorca 2 (ang. Compute 2):
   CPU: 2.5GHz (4 rdzenie)
   RAM: 16GB
   HDD: 160GB
   interfejsy: 2xGbE; eth0 (sieć dostępowa), eth1 (sieć wewnętrzna)
   system operacyjny: CentOS 7 (64bit)

Zaczynamy!

Na wszystkich węzłach chmury, tj. Kontroler, Hipernadzorca 1, Hipernadzorca 2, wyłączamy aplikację NetworkManager w celu uniknięcia ewentualnych przykrych niespodzianek z połączeniem sieciowym na poszczególnych węzłach:

```
$ systemctl stop NetworkManager
$ systemctl disable NetworkManager
```

Instalację Packstacka uruchomimy z Kontrolera chmury (ang. Controller Node), w tym celu instalujemy repozytorium *centos-release-openstack-pike* na Kontrolerze:

```
$ yum install centos-release-openstack-pike
```

Aktualizujemy Kontroler w celu pobrania najnowszych niezbędnych do instalacji paczek z repozytorium *centos-release-openstack-pike:*

```
$ yum update
```

Instalujemy Packstacka:

```
$ yum install openstack-packstack
```

Plik konfiguracyjny Packstacka

Packstack do instalacji potrzebuje pliku konfiguracyjnego (ang. Answer File) i jeśli do instalacji typu All-In-One (wszystkie serwisy na jednym serwerze) przygotowywanie takiego pliku nie jest wymagane, to w przypadku instalacji na trzech serwerach/węzłach taki plik będzie niezbędny do prawidłowego wdrożenia chmury.

Wstępny szablon pliku konfiguracyjnego *answers.txt* możemy wygenerować z pomocą Packstacka:

```
$ packstack
--gen-answer-file=/root/answers.txt
```

Szablon po wygenerowaniu zawiera większość domyślnie ustawionych parametrów, część z nich musimy jednak zdefiniować lub zmodyfikować. Listing 1 zawiera listę kluczowych parametrów, które zdefiniowaliśmy dla naszej przykładowej instalacji.

Pozostałe parametry, niewymienione w powyższym listingu, pozostawiamy z ich domyślnymi wartościami.

Instalacja Chmury

Kiedy już mamy przygotowany plik konfiguracyjny, czas rozpocząć instalację. W celu uniknięcia błędów limitów czasowych (ang. timeout) związanych z wykonywaniem poszczególnych procedur Puppeta, będącego główną siłą napędową Packstacka, zdecydowaliśmy się zwiększyć limit czasowy do 600 sekund, podczas gdy domyślna wartość wynosi 300 sekund.

Rozpoczynamy zatem instalację z poziomu Kontrolera:

```
$ packstack
--answer-file=/root/answers.txt
--timeout=600
```

Instalacja na ww. serwerach trwa około 45 minut, czas ten może się znacznie wydłużyć, w zależności od użytego sprzętu.

OPIS WYBRANYCH PARAMETRÓW PLIKU *ANSWERS.TXT*

CONFIG_HEAT_INSTALL – opcja instalacji usługi Heat Orchestration

CONFIG_NTP_SERVERS – lista serwerów NTP (Network Time Protocol)

CONFIG_CONTROLLER_HOST – adres IP kontrolera chmury

CONFIG_COMPUTE_HOSTS – lista adresów IP hipernadzorców

CONFIG_NETWORK_HOSTS – lista adresów IP węzłów/agentów sieciowych

CONFIG_KEYSTONE_ADMIN_USERNAME – login administratora chmury

CONFIG_KEYSTONE_ADMIN_PW – hasło administratora chmury

CONFIG_NEUTRON_L3_EXT_BRIDGE – nazwa mostka sieciowego do komunikacji zewnętrznej

CONFIG_NEUTRON_ML2_TYPE_DRIVERS – lista sterowników dla wybranych typów sieci, które mają być obsługiwane przez chmurę, a konkretnie usługę Neutron

CONFIG_NEUTRON_ML2_TENANT_NETWORK_TYPES – lista dozwolonych typów sieci dla tworzonych projektów/tenantów wewnątrz chmury

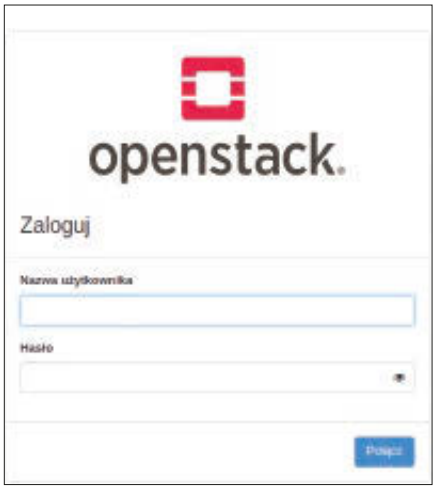
CONFIG_NEUTRON_ML2_VLAN_RANGES – zakres tagowania sieci typu VLAN tworzonych wewnątrz chmury. Dla wybranego zakresu VLAN należy upewnić się, że porty na przełącznikach, do których podpięte są interfejsy eth1 poszczególnych węzłów chmury, są odpowiednio skonfigurowane (otagowane), aby pozwolić na ruch sieciowy dla danego zakresu.

CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS – wybór mostka sieciowego, obsługującego ruch w sieciach VLAN

CONFIG_NEUTRON_OVS_BRIDGE_IFACES – mapowanie interfejsów fizycznych z mostkami sieciowymi OVS

CONFIG_NEUTRON_OVS_BRIDGES_COMPUTE – lista mostków OVS, które zostaną utworzone na potrzeby komunikacji VLAN pomiędzy hipernadzorcami

Prawidłowo wykonana instalacja objawi się nam komunikatem zawierającym adres panelu administracyjnego (ang. Dashboard) oraz lokalizację dzienników z instalacji (Listing 2).



Rysunek 2: Ekran powitalny panelu administracyjnego Horizon.

Wstępny test poprawności instalacji

Przeprowadźmy krótki test poprawności naszej instalacji. W tym celu wpisujemy w przeglądarce internetowej adres panelu administracyjnego podany w komunikacie poinstalacyjnym:

```
http://192.168.2.60/dashboard
```

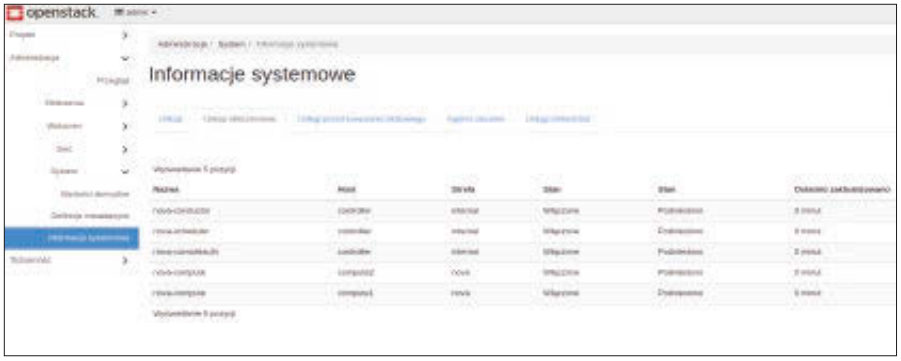
Naszym oczom powinien ukazać się ekran powitalny OpenStacka (Rysunek 2).

Logujemy się, używając danych logowania z pliku konfiguracyjnego, czyli w naszym przypadku: *admin/password*.

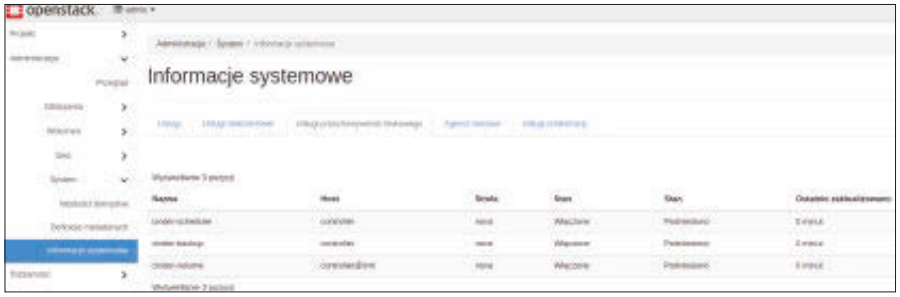
Po zalogowaniu w menu wybieramy zakładkę *Administracja*, następnie *Informacje systemowe* i sprawdzamy kolejno zakładki:

- *Usługi obliczeniowe* (Rysunek 3)
- *Usługi przechowywania blokowego* (Rysunek 4)
- *Agenci sieciowi* (Rysunek 5)
- *Usługi orkiestracji* (Rysunek 6)

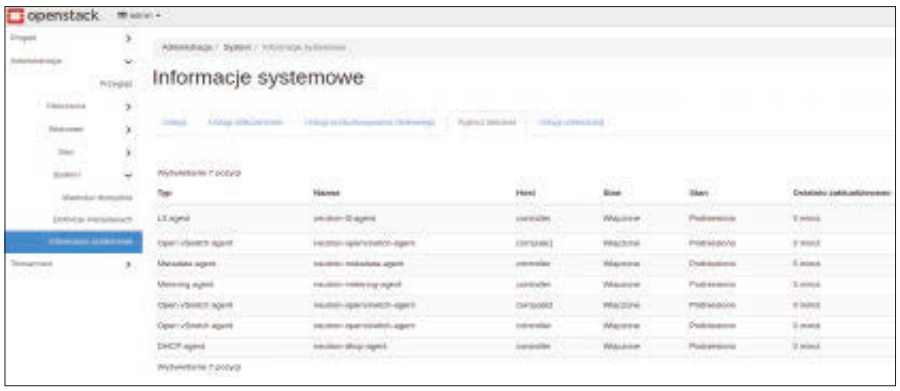
Jeśli chmura została zainstalowana poprawnie, wszystkie usługi powinny wskazywać stan: *Podniesiono*.



Rysunek 3: Usługi obliczeniowe w chmurze OpenStack.



Rysunek 4: Usługi przechowywania blokowego w chmurze OpenStack.



Rysunek 5: Usługi sieciowe działające na poszczególnych węzłach chmury OpenStack.

Plik uwierzytelniający Keystone

Po zakończeniu instalacji skrypt Packstack tworzy domyślnie w katalogu */root* plik uwierzytelniający administratora

LISTING 1: Kluczowe parametry pliku konfiguracyjnego *answers.txt*

```
CONFIG_HEAT_INSTALL=y
CONFIG_NTP_SERVERS=time1.google.
com,time2.google.com
CONFIG_CONTROLLER_HOST=192.168.2.60
CONFIG_COMPUTE_
HOSTS=192.168.2.29,192.168.2.30
CONFIG_NETWORK_HOSTS=192.168.2.60
CONFIG_KEYSTONE_ADMIN_
USERNAME=admin
CONFIG_KEYSTONE_ADMIN_PW=password
CONFIG_NEUTRON_L3_EXT_BRIDGE=br-ex
CONFIG_NEUTRON_ML2_TYPE_
DRIVERS=vxlan,vlan,flat
CONFIG_NEUTRON_ML2_TENANT_NET-
WORK_TYPES=vlan
CONFIG_NEUTRON_ML2_VLAN_
RANGES=physnet1:1200:1209
CONFIG_NEUTRON_OVS_BRIDGE_
MAPPINGS=physnet1:br-eth1
CONFIG_NEUTRON_OVS_BRIDGE_
IFACES=br-eth1:eth1
CONFIG_NEUTRON_OVS_BRIDGES_
COMPUTE=br-eth1
```

LISTING 2: Końcowy komunikat instalatora Packstack

```
**** Installation completed successfully
******

Additional information:
 * File /root/keystonerc_admin has been
created on OpenStack client host 192.168.2.60.
To use the command line tools you need to
source the file.
 * To access the OpenStack Dashboard browse
to http://192.168.2.60/dashboard .
Please, find your login credentials stored in
the keystonerc_admin in your home direc-
tory.
 * Because of the kernel update the host
192.168.2.30 requires reboot.
 * Because of the kernel update the host
192.168.2.29 requires reboot.
 * The installation log file is available at: /var/
tmp/packstack/20171209-213413-tOCFn_/
openstack-setup.log
 * The generated manifests are available at: /
var/tmp/packstack/20171209-213413-tOCFn_/
manifests
```

Rysunek 6: Usługa orkiestracji działająca na Kontrolerze chmury OpenStack.

LISTING 3: Zawartość pliku *keystonerc_admin*

```
unset OS_SERVICE_TOKEN
export OS_USERNAME=admin
export OS_PASSWORD='password'
export OS_AUTH_
URL=http://192.168.2.60:5000/v3
export PS1='[\u@\h \W(keystone_admin)]\$ '
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_
NAME=Default
export OS_IDENTITY_API_VERSION=3
```

keystonerc_admin, zawierający m.in. dane uwierzytelniające administratora chmury, tj. login, hasło (Listing 3).

Zmienne zawarte w pliku możemy wyeksportować jako zmienne środowiskowe naszej sesji poleceniem:

```
$ source /root/keystonerc_admin
```

Od tej pory jesteśmy uwierzytelnieni na Kontrolerze chmury jako użytkownik *admin*, co daje nam możliwość zarządzania chmurą z poziomu wiersza poleceń.

Będąc uwierzytelnieni jako administrator chmury, możemy przykładowo sprawdzić wspomniane wyżej usługi działające w chmurze z poziomu wiersza poleceń:

Wyświetlenie listy hipernadzorców chmury:

```
(keystone_admin)$ nova hypervisor-list
```

Wyświetlanie usług działających w chmurze:

```
(keystone_admin)$ nova service-list
```

Wyświetlanie listy agentów sieciowych:

```
(keystone_admin)$ openstack network agent list
```

Konifguracja po instalacji

W założeniach niniejszego artykułu przyjęliśmy, że interfejsy *eth0* na poszczególnych węzłach będą podłączone do głównej sieci dostępowej naszej chmury (*provider_net*). Dla uproszczenia konfiguracji sieciowej będą on pełniły także funkcję interfejsu dostępowego do poszczególnych instancji (maszyn wirtualnych) działających wewnątrz chmury OpenStack. Dostęp do instancji w chmurze uzyskamy, łącząc sieć dostępową poprzez tzw. *qrouter* – wirtualny router działający w chmurze, z wewnętrzną siecią, do której będą podłączone interfejsy instancji wewnątrz projektu, zwanego również najemcą (ang. tenant). Każda instancja otrzyma zatem adres IP w sieci zewnętrznej, tzw. *Floating IP*, mapowany na jej adres IP w sieci wewnętrznej w celu zarządzania nią z zewnątrz. Innymi słowy, użyjemy tej samej sieci do zarządzania węzłami chmury, jak też instancjami działającymi wewnątrz niej. Rozwiązanie takie nie jest używane w systemach produkcyjnych, chociażby ze względu na niebezpieczeństwo, że klienci takiej chmury będą mieli dostęp do interfejsów zarządzania samą chmurą. Niemniej jednak do celów demonstracyjnych w izolowanym środowisku testowym możemy przyjąć takie uproszczenie.

Interfejsy *eth1* na poszczególnych węzłach chmury będą używane do komunikacji wewnętrznej między instancjami. Będzie ona miała znaczenie, w przypadku gdy np. w jednym projekcie (tenancie) będzie utworzonych kilka instancji, z których każda fizycznie będzie działała na innym hipernadzorcy.

Interfejsy *eth1* zostały przez Packstacka już podpięte na każdym hipernadzorcy jako porty typu OVS do mostka *br-eth1*, który z kolei podpięty jest do *br-int*, czyli mostka integracyjnego, spinającego wszystkie instancje na danym hipernadzorcy (Rysunek 7).

Możemy tę informację wyświetlić w konsoli dowolnego hipernadzorcy:

LISTING 4: Plik konfiguracyjny *ifcfg-br-ex*

```
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
NAME=br-ex
DEVICE=br-ex
ONBOOT=yes
IPADDR=192.168.2.60
PREFIX=24
GATEWAY=192.168.2.1
DNS1=8.8.8.8
DNS2=8.8.4.4
NM_CONTROLLED=no
```

```
$ ovs-vsctl list-ifaces br-eth1
eth1
phy-br-eth1
```

Zawartość pliku *ifcfg-eth1* po modyfikacjach Packstack wygląda następująco:

```
DEVICE=eth1
NAME=eth1
DEVICETYPE=ovs
TYPE=OVSPort
OVS_BRIDGE=br-eth1
ONBOOT=yes
BOOTPROTO=none
```

Pozostaje nam zatem dodanie interfejsu *eth0* jako portu do mostka sieciowego *br-ex* (ang. Bridge External) w celu utworzenia zewnętrznego interfejsu dostępowego dla chmury. Tym sposobem interfejs *br-ex* otrzyma adres IP interfejsu *eth0*. Procedurę tę wykonujemy tylko na kontrolerze chmury.

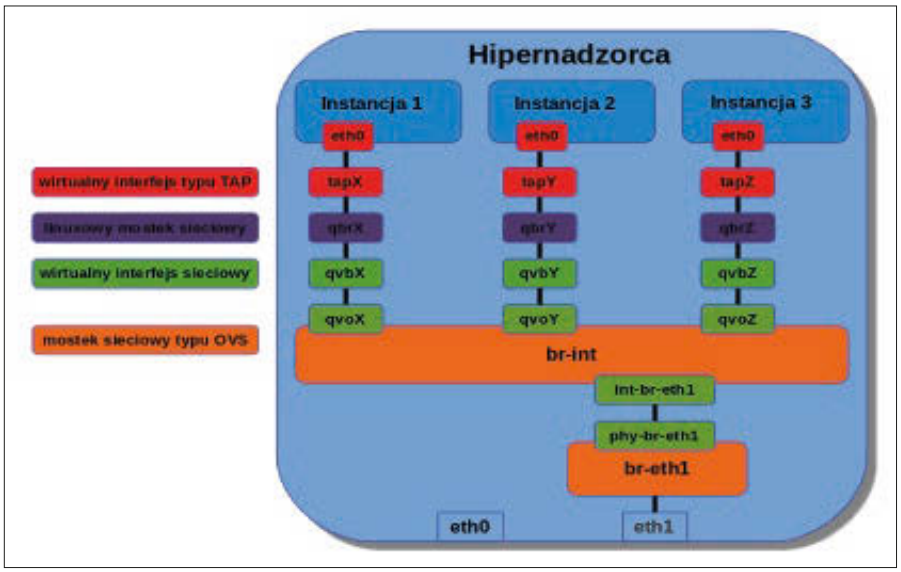
Obecnie adres IP kontrolera przypisany jest do interfejsu *eth0*:

```
2: eth0: <BROADCAST,MULTICAST,UP,
LOWER_UP>
 mtu 1500 qdisc pfifo_fast state UP qlen 1000
   link/ether 52:54:00:cc:72:eb brd ff:ff:ff:ff:ff:ff
   inet 192.168.2.60/24 brd 192.168.2.255
scope global eth0
    valid_lft forever preferred_lft forever
   inet6 fe80::5054:ff:fecc:72eb/64 scope link
    valid_lft forever preferred_lft forever
```

Naszym celem jest przeniesienie adresu IP kontrolera z interfejsu *eth0* na interfejs *br-ex*.

Robimy kopię zapasową pliku *ifcfg-eth0* przed jego modyfikacją:

```
$ cp /etc/sysconfig/network-scripts/ifcfg-eth0
/root/ifcfg-eth0.backup
```



Rysunek 7: Schemat architektury sieciowej hipernadzorcy (Compute node) dla sieci wewnętrznych typu VLAN, zintegrowanej z wirtualnym przełącznikiem Open vSwitch (OVS).

Następnie kopiujemy plik *ifcfg-eth0* na plik *ifcfg-br-ex*, tworząc tym samym wstępną konfigurację dla interfejsu *br-ex*:

```
$ cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-br-ex
```

Modyfikujemy plik *ifcfg-eth0* w taki sposób, aby stał się portem typu OVS dla mostka *br-ex*:

```
DEVICE=eth0
ONBOOT=yes
DEVICETYPE=ovs
TYPE=OVSPort
OVS_BRIDGE=br-ex
```

W nowo utworzonym pliku *ifcfg-br-ex* zamieniamy interfejs *eth0* na *br-ex* (Listing 4).

Podpinamy interfejs *eth0* jako port do mostka *br-ex*. Poniższa komenda spowoduje chwilowe rozłączenie kontrolera chmury od sieci, jeśli więc jesteśmy podłączeni do kontrolera za pomocą protokołu SSH, stracimy na chwilę połączenie, które powinno zostać po kilku sekundach przywrócone, jeśli poprawnie skonfigurowaliśmy oba interfejsy:

```
$ ovs-vsctl add-port br-ex eth0; systemctl restart network
```

Po naszych modyfikacjach mostek *br-ex* posiada teraz port typu OVS w postaci *eth0*:

```
$ ovs-vsctl list-ifaces br-ex
eth0
```

Adres IP kontrolera został przeniesiony z interfejsu *eth0* na mostek *br-ex*:

```
6: br-ex: <BROADCAST,MULTICAST,UP, LOWER_UP>
 mtu 1500 qdisc noqueue state UNKNOWN qlen 1000
    link/ether 16:6c:d9:c4:7b:4d brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.60/24 brd 192.168.2.255 scope global br-ex
       valid_lft forever preferred_lft forever
    inet6 fe80::146c:d9ff:fec4:7b4d/64 scope link
       valid_lft forever preferred_lft forever
```

Na koniec warto stworzyć wewnątrz infrastruktury OpenStack zewnętrzną współdzieloną sieć dostępową *provider_net*, o której wspominaliśmy już wcześniej w niniejszym artykule. Sieci wewnętrzne poszczególnych projektów/tenantów będą podłączone do tej sieci przez wspomniany wcześniej *qrouter*. W tym celu, jeżeli nie jesteśmy w tym momencie uwierzytelnieni jako *admin*, musimy ponownie wyeksportować zmienne środowiskowe użytkownika *admin* poleceniem:

```
$ source /root/keystonerc_admin
```

Następnie tworzymy sieć zewnętrzną *provider_net* poleceniem:

```
(keystone_admin)$ openstack network create --external --share provider_net
```

...oraz powiązaną z nią podsieć *provider_subnet* zawierającą jej parametry (maskę, bramę, pulę adresów IP) poleceniem:

```
(keystone_admin)$ openstack subnet create --subnet-range 192.168.2.0/24 --no-dhcp --gateway 192.168.2.1 --network provider_net provider_subnet
```

Sieć *provider_net* pozwoli nam na dostęp z zewnątrz do instancji działających wewnątrz chmury poprzez pulę adresów Floating IP.

To wszystko! Właśnie zainstalowaliśmy i skonfigurowaliśmy środowisko OpenStack w wersji Pike oparte na trzech serwerach. Oczywiście to nie koniec rozszerzalności chmury, możemy do niej dodawać kolejne węzły, a w szczególności hipernadzorców. Chmurę możemy rozszerzać za pomocą Packstacka, w trybie on-line, a więc bez jakichkolwiek przestojów w jej działaniu, nie jest to jednak tematem niniejszego artykułu.

Podsumowanie

OpenStack, mimo że składa się praktycznie w większości ze znanych, sprawdzonych, a czasem nawet wysłużonych technologii, takich jak chociażby KVM czy MariaDB, jako całość stanowi w dalszym ciągu nową, dobrze rokującą i prężnie rozwijającą się technologię, która praktycznie z każdą wersją zyskuje coraz więcej dodatków i funkcji, a co za tym idzie, również użytkowników. W niniejszym artykule zaprezentowaliśmy, w jaki sposób zainstalować przykładowe środowisko RDO OpenStack w wersji Pike do celów demonstracyjnych, czyli tzw. proof of concept. Nasza chmura jest zatem gotowa na „przyjęcie pierwszych najemców". Innymi słowy, możemy już w niej tworzyć pierwsze projekty, dodawać użytkowników, ustawiać limity wykorzystania zasobów (ang. quota), tworzyć sieci wewnątrz tenantów i wreszcie uruchamiać instancje.

AUTOR

Grzegorz Juszczak – inżynier systemowy w dziale utrzymania infrastruktury 5G w firmie Nokia Solutions and Networks Sp. z o.o. Entuzjasta systemów linuksowych i OpenStacka. Założyciel bloga technicznego http://www.tuxfixer.com o tematyce Linuxa, wirtualizacji i chmur obliczeniowych.

INFO

[1] Dokumentacja OpenStack Pike: https://docs.openstack.org/pike/
[2] Strona projektu RDO: https://www.rdoproject.org/

Promocja otwartego oprogramowania

Jak sprzedać otwarte oprogramowanie

Marketing w przypadku FOSS, w porównaniu z własnościowym oprogramowaniem, wymaga nowatorskiego podejścia. Dzielimy się naszymi doświadczeniami. Mike Saunders

Wiele już razy słyszeliśmy, że „otwarte oprogramowanie, takie jak Linuks, nie potrzebuje reklamy!". W jakimś sensie jest to prawda: ludzie dowiadują się o FOSS najczęściej od znajomych. Wielu z nas odkryło Linuksa u przyjaciół, kolegów czy czytając o nim w dyskusjach w sieci, a nie z reklam w telewizji. Używamy go, ponieważ się nam podoba, a nie dlatego że zostaliśmy zmanipulowani. Będziemy z niego korzystać, promować i wspierać go, bez względu na to, co się stanie z firmami, które go tworzą czy używają.

Z drugiej strony, Linuks i ogólnie FOSS istnieją na wysoce konkurencyjnym (i nieprzyjemnym) rynku. Jeśli chcemy, aby udziały wolnego oprogramowania rosły, musimy rozważyć użycie strategii marketingowych stosowanych przez konkurencję. Powinniśmy zwalczać „strach, niepewność i wątpliwości" rozsiewane przez ludzi źle życzących FOSS. Chcemy przedstawiać oprogramowanie, które kochamy, w pozytywnym świetle, nawet jeśli nie wszystko idzie po naszej myśli (jak w przypadku zagrożenia bezpieczeństwa nazwanego Heartbleed).

Reklamowanie otwartego oprogramowania to coś, co wszyscy możemy zrobić, korzystając z mediów społecznościowych, tworząc nagrania i infografiki wyjaśniające, czym jest FOSS, lub uczęszczając na spotkania i dając wykłady. Na kolejnych stronach przeanalizujemy niektóre klasyczne sposoby na promocję oprogramowania, przyjrzymy się przykładowi Firefoksa oraz IBM i podamy kilka rad, z których możecie skorzystać.

Otwarte vs. zamknięte

Wyobraźmy sobie, że jesteśmy nowo powołanymi menedżerami marketingu w firmie zajmującej się własnościowym oprogramowaniem. Jakie są nasze obowiązki? Zaczniemy pewnie od sprawdzenia aktualnego portfolio produktów i ustalimy, jak wpasowują się one w rynek (czyli do kogo są skierowane, czy jest szansa na ich rozwój i czy ich wycena jest prawidłowa). Przyjrzymy się też produktom w fazie rozwoju, aby dowiedzieć się, co nas czeka. Z czasem zyskamy doświadczenie i zaczniemy ściślejszą współpracę z deweloperami, pomagając im kształtować oprogramowanie.

Wyobraźmy sobie teraz, że mamy to samo stanowisko, ale w projekcie wolnego oprogramowania (lub w firmie sprzedającej FOSS). Niektóre z naszych obowiązków będą takie same, ale w przypadku innych, różnice będą drastyczne. Na przykład, kiedy otwartoźródłowy program jest tworzony przez dużą społeczność, nie da się w łatwy sposób pokierować jego rozwojem. Oczywiście, możemy zbadać rynek i określić, czego ludzie potrzebują, ale co dalej? Podajemy te informacje naszej społeczności i niewiele z tego wynika. Trudno jest ukierunkować deweloperów, którzy tworzą coś, czego potrzebują, i nie są przez nas opłacani.

Podobnie jest, jeśli sprzedajemy nie samo oprogramowanie, lecz wsparcie i inne usługi z nim związane (jak wiele podobnych firm na rynku FOSS). Nasze podejście musi być inne. Poza tym, nie tylko chcemy, aby użytkownik końcowy wypróbował nasz produkt, ale też potrzebujemy zaangażowania potencjalnych współtwórców. Nasza praca wymaga więc dobrego kontaktu i zarządzania społecznością.

Klasyczny Przykład nr 1: IBM

W późnych latach 90. XX w. GNU/Linux był w miarę przyzwoitym systemem operacyjnym dla serwerów, zyskiwał klientów wśród małych firm i dostawców usług internetowych, ale brakowało mu większego komercyjnego wsparcia. Było już kilka firm w ekosystemie FOSS wspierających go, takich jak Red Hat, ale były to raczej płotki w tamtych czasach. Linux był domeną geeków i hakerów. Brakowało dobrego wizerunku i pieniędzy, aby zmienić sytuację.

Było tak, do czasu, kiedy w FOSS zaangażował się IBM. W 2000 roku firma ogłosiła, że na

przestrzeni roku wyda miliard dolarów na rozwój Linuksa [1]. Powód? Louis Gerstner, dyrektor generalny IBM w tym czasie, stwierdził, że firma „jest przeświadczona, że Linuks może być dla aplikacji biznesowych tym, czym Internet dla sieci i komunikacji". IBM posiadał już wtedy całą gamę systemów operacyjnych, wliczając w to pochodną Uniksa, zwaną AIX, a mimo to poważnie zaangażował się w rozwój naszej otwartoźródłowego pisklęcia.

Ofiarowany miliard dolarów wydany został na usprawnianie GNU/Linuksa i innych narzędzi FOSS, a jednocześnie IBM rozpoczął promowanie nowego systemu operacyjnego. Kampania ta wydała na przykład 90-sekundową reklamę nazwaną „Prodigy" (Cudowne dziecko), którą ciągle można znaleźć na YouTubie [2]. W filmie tym mały chłopiec siedzi samotnie w białym pokoju (Rysunek 1).

W ciągu 90 sekund różni sławni ludzie podchodzą do chłopca i udzielają mu rad na temat życia – pojawił się nawet Muhammad Ali, radząc: „Mów to, co myślisz. Nie wycofuj się". W tym samym czasie dwie niezidentyfikowane postaci robią notatki. Na końcu filmu jedna z nich pyta drugą: „Jak on ma na imię?". Pada odpowiedź: „Ma na imię Linux".

Na tym wideo się kończy. Jest frapujące i twórcze, pokazuje nowatorskie podejście do reklamy oprogramowania. IBM nie bał się przedstawić Linuksa jako kogoś młodego i raczej niedoświadczonego, ale skupił się na jego możliwościach uczenia się i przyswajania informacji (poprzez społeczność wolnego oprogramowania).

Co jednak szczególnie zasługuje na uwagę, to pozycjonowanie tej reklamy. Nie był to branżowy żart dla geeków pokazywany na konferencjach. Reklama ta ukazała się w czasie Super Bowl w 2005 roku, wydarzenia z jedną z największych publiczności telewizyjnych na świecie. IBM robił więcej, niż wydawało się to niezbyt inteligentnym szefom IT. Pokazywał wszystkim, że Linuks to poważna sprawa.

Klasyczny Przykład Numer 2: Firefox

IBM skupiał się na wdrażaniu Linuksa w dużych przedsiębiorstwach. Jeśli chcemy przykładu promocji skierowanej do konsumentów, możemy sięgnąć do przykładu Fundacji Mozilla i jej spektakularnego wejścia na rynek w 2004 roku. Wydano właśnie Firefoksa 1.0, Internet Explorer ciągle dominował, a nowa przeglądarka miała ukazać ludziom świat FOSS, w idealnej sytuacji zachęcając ich do przyjrzenia się wolnemu oprogramowaniu, a nawet wypróbowania Linuksa.

Fundacja Mozilla stworzyła stronę „Spread Firefox" (Rozprzestrzeniaj Firefoksa) i pozyskała fundusze, które można było wykorzystać na promocję. W rezultacie powstała dwustronicowa reklama w *New York Times* (Rysunek 2), która

Rysunek 1: Reklama IBM z 2004 roku, nazwana „Prodigy", przedstawiła Linuksa milionom ludzi w USA.

zadawała czytelnikom pytanie, czy nie mają dosyć swoich przeglądarek, a następnie pokazywała alternatywę. Dalej przedstawiono cytaty z wypowiedzi użytkowników, troszkę informacji na temat aplikacji i adres, pod którym można ją znaleźć.

W rezultacie, w ciągu kolejnych 18 miesięcy Firefox potroił udziały w rynku, przekraczając 10 procent w czerwcu 2006 roku (szczytem było 32 procent w 2009 roku). Oczywiście wpływ na to miały też inne czynniki, ale jest to wspaniały przykład tego, jak w prosty, przemyślany i efektywny sposób projekt FOSS może uzyskać szerszą publiczność.

Ryzykowne podejście

Reklamując i wychwalając otwartoźródłowe oprogramowanie, bardzo łatwo jest ulec pokusie, tworząc zbyt odważne stwierdzenia czy daleko idące uogólnienia. Jak wiele razy słyszeliście, że „otwarte oprogramowanie jest bezpieczniejsze niż własnościowe"? Wiemy, że coś w tym jest. Historia ciągle od nowa pokazuje, że własnościowe

Rysunek 2: Mozilla zwróciła się do zdegustowanych użytkowników IE za pomocą dwustronicowej reklamy w *New York Times*.

Rysunek 3: Bądźmy ostrożni, reklamując FOSS za pomocą argumentu „jest bardziej bezpieczny”, gdyż zawsze może się zdarzyć coś takiego jak Heartbleed.

aplikacje często pełne są dziur bezpieczeństwa wykorzystywanych przez niezbyt miłych ludzi przez lata, zanim zostaną ujawnione.

Musimy jednak być ostrożni. Bywa, że powiemy, że „otwarte oprogramowanie jest bezpieczne”, i wtedy zdarza się coś takiego jak Heartbleed [3] (Rysunek 3). Jeśli nie słyszeliście, to Heartbleed był poważną luką w zabezpieczeniach w bibliotece OpenSSL, z której korzystają praktycznie wszyscy. Błąd ujawniono w 2014 roku i miało to wpływ na dużą liczbę stron. Było źle do tego stopnia, że ludzie od OpenBSD stworzyli odgałęzienie biblioteki, które wyczyścili i nazwali LibreSSL.

Co jednak najważniejsze: ci z nas, którzy powtarzali jak mantrę, że „otwarte oprogramowanie jest bezpieczniejsze” wyszli na błaznów. Oczywiście, dalej wierzymy, że to zdanie ogólnie jest prawdziwe, ale Heartbleed dostarczył przeciwnikom poważnych argumentów. „Otwarte aplikacje również są niebezpieczne”, „otwarte oprogramowanie posiada luki, których przez lata nikt nie łata” i tak dalej.

Dlatego ważne jest, aby stwierdzać, że otwarte oprogramowanie samo w sobie nie jest w magiczny sposób bezpieczniejsze. Liczy się proces jego tworzenia. Można też zauważyć, że OpenSSL nie był do końca otwarty. Tak, kod jest dostępny, ale mało kto nad nim pracował. Ci, którzy zajrzeli do środka, uciekali z krzykiem. Jak widać, często powtarzane zdanie: „Wiele par oczu, to mało błędów” nie miało tutaj zastosowania.

Mając to wszystko na uwadze, nie powinniśmy mówić, że „otwarte aplikacje są bardziej bezpieczne”, ale sprecyzować, że „proces tworzenia otwartego oprogramowania sprzyja tworzeniu bezpieczniejszych aplikacji”. Nie jest to aż tak chwytliwe zdanie, rodzi więcej pytań, ale jest uczciwe. To samo podejście zastosujmy w dziedzinie niezawodności. Wiemy, że GNU/Linux to całkiem solidny system, ale musimy pamiętać, że są użytkownicy, którzy stykają się z rzadziej występującymi błędami. Chociaż żarty z Microsoftu i ich „niebieskiego ekranu śmierci” mogą być kuszące, to pamiętajmy, że już nie konkurujemy z Windowsem ME. Windows ma wiele problemów, jesteśmy szczęśliwi, że my ich unikamy, ale dla wielu użytkowników jest całkiem solidny.

Kasa, kasa, kasa

Inny obszar, w którym należy być ostrożnym, to finanse. Owszem, GNU/Linux jest darmowy, ale są też inne czynniki, które determinują koszt systemu, na przykład jeśli musimy wymienić sprzęt na coś, co jest kompatybilne z Linuksem i co winduje ogólny koszt. W przypadku kiedy chcielibyśmy wprowadzić Linuksa (lub inny rodzaj otwartego oprogramowania, na przykład LibreOffice) w dużej firmie, konieczne może być przeszkolenie użytkowników końcowych. To również podnosi koszt.

Dlatego wiele osób wykonujących zakupy w branży IT, mówi o całkowitym koszcie posiadania (TCO – total cost of ownership). Jak wiele będzie kosztowało użytkowanie programu X, biorąc pod uwagę zmiany w sprzęcie, szkolenia i wsparcie? W krótkoterminowej perspektywie nie wygląda to dobrze dla Linuksa i FOSS. Jeśli duża firma posiada już własnościowy system, mając prawdopodobnie zniżki na jego zakup, wtedy zmiana na rozwiązanie FOSS może być początkowo bardzo kosztowna.

Weźmy to pod uwagę, reklamując FOSS. Nie mówmy: „Linux jest darmowy”, ponieważ praktycznie każde większe wdrożenie wymaga jakiegoś rodzaju wsparcia czy od Red Hata, SUSE, czy Canonicala lub firmowego zespołu IT. Lepiej jest powiedzieć: „W dłuższym okresie Linux/FOSS może zredukować całkowity koszt użytkowania oprogramowania, gdyż zwiększa niezawodność, bezpieczeństwo i nie wymaga opłat licencyjnych”. Możemy uściślić, co rozumiemy przez niezawodność i bezpieczeństwo. To jest właśnie odpowiedni sposób sprzedaży FOSS.

Jeszcze na tamat wsparcia – dobrym chwytem marketingowym jest podkreślanie, że w przypadku FOSS mamy więcej możliwości wsparcia. Przykładowo, jeśli firma posiada tysiące pecetów z pakietem Microsoft Office, a poważny błąd wpływa na pracę wielu użytkowników, to co można zrobić? Należy zadzwonić do Microsoftu, prawdopodobnie wydać więcej pieniędzy i mieć nadzieję, że błąd zostanie naprawiony z kolejną aktualizacją. Jeśli nie, to w sumie nie ma już zbyt wielu możliwości działania.

Dla kontrastu, rozważmy firmę posiadającą pakiety LibreOffice'a. Jeśli błąd wpływa na pracę użytkowników, firma może wybrać jednego z wielu certyfikowanych deweloperów [4] i zatrudnić tego, który (miejmy nadzieję) go naprawi. Oczywiście to również kosztuje, ale firma nie jest na łasce jednego dostawcy i może zbadać rynek, szukając rozwiązania (nie wykluczając lokalnych deweloperów). W tym wypadku zadziała wolny rynek. Piękny argument dla powtarzających, że FOSS jest antykapitalistyczny!

Co możemy zrobić?

Wiele projektów FOSS wręcz rozpaczliwie potrzebuje reklamy. Czasami mają fatalnie zrobione strony, bez dobrze podanej informacji na temat tego, co właściwie robi aplikacja (zaskakująco często spotykana sytuacja). W innych przypadkach projekt idzie do przodu, ale nie

Rysunek 4: Społeczność zajmująca się marketingiem w Mozilli ma nawet własną maskotkę.

informuje o tym nikogo. Możliwe, że dane oprogramowanie mogłoby uzyskać nowych użytkowników przez media społecznościowe, Reddita czy Hacker News, ale żaden ze współtwórców nie wie, jak się do tego zabrać.

Oczywiście, wiele projektów FOSS nie ma budżetów pozwalających na zatrudnienie osoby do marketingu, dlatego jeśli interesuje nas ten temat, warto wyciągnąć rękę do deweloperów. Nie musimy być ekspertami w danej dziedzinie. Każda pomoc się przyda. Na rynku istnieje wiele książek, dzięki którym zapoznamy się z marketingiem. Z podstawowymi umiejętnościami i technikami możemy wesprzeć otwarte projekty, których używamy i chcemy, aby prosperowały.

Warto też przyjrzeć się, jak to robią duże projekty FOSS. Przykładowo społeczność Mozilli stworzyła marketingowy przewodnik [5] oraz wiele materiałów w formie stron i dokumentów [6] (Rysunek 4). Fedora zaś posiada zespół marketingowy skupiony wokół listy e-mailingowej [7], a w LibreOffice istnieje mała społeczność pracująca nad prezentacjami, konferencjami prasowymi i innymi materiałami (Rysunek 5), organizująca raz w miesiącu telekonferencje [8].

Dodatkowym plusem jest to, że dzięki wsparciu uzyskamy doświadczenie, które zaprocentuje na rynku pracy. Jeśli dzięki temu któregoś dnia kupimy sobie prywatną wyspę, nie zapomnijmy też o skromnym autorze z *Linux Magazine*, który nakierował nas na tę ścieżkę. ■■■

INFO

[1] IBM i Linux: https://www.cnet.com/news/ibm-to-spend-1-billion-on-linux-in-2001/
[2] Reklama IBM „Prodigy”: https://www.youtube.com/watch?v=s7dTjpvakmA
[3] Heartbleed: http://heartbleed.com
[4] Certyfikowani dewloperzy LibreOffice: http://www.documentfoundation.org/gethelp/developers/
[5] Przewodnik marketingowy Mozilli: https://wiki.mozilla.org/MarketingGuide
[6] Materiały marketingowe Mozilli: https://wiki.mozilla.org/Marketing:Firefox_Materials
[7] Marketing w Fedorze: https://fedoraproject.org/wiki/Marketing
[8] Marketing w LibreOffice: https://wiki.documentfoundation.org/Marketing

Rysunek 5: Inspiracje mogą przynieść istniejące projekty FOSS. Przykładem mogą być ulotki wydawane przez zespół LibreOffice'a.

Z pomocą Audacity digitalizujemy muzykę z naszych płyt i kaset

Remastering

Uzbrojeni w edytor dźwięku Audacity, możemy przenieść analogową zawartość płyt i kaset do świata cyfrowego. Mario Blaettermann

Stare dobre płyty winylowe przeżywają drugą młodość (hipsterzy wracają nawet do kaset magnetofonowych). Jednak te skarby przeszłości nie znoszą zbyt dobrze upływu czasu. Dzięki Audacity zarchiwizujemy na dysku twardym, w formie cyfrowej, naszą analogową muzykę.

Era analogowa

W swoim czasie płyty oraz kasety były kamieniami milowymi rozwoju przemysłu muzycznego. Wiele osób do dzisiaj posiada gdzieś w piwnicy czy na strychu półki pełne tych artefaktów. Chociaż jednak płyty winylowe wracają do łask, to miniaturowe wersje taśm szpulowych powoli odchodzą w niebyt.

W większości przypadków te dawne skarby zastępowane są płytami CD, ale niektóre starsze nagrania, utworzone w profesjonalnych studiach nagraniowych, są dostępne jedynie na płytach winylowych. Prywatne nagrania, dema czy utwory szkolnych zespołów są związane właśnie ze starymi nośnikami. Dlatego ważne jest, aby ochronić je przez zniszczeniem.

Profesjonalne podejście

Najprostszym sposobem na digitalizację naszych materiałów jest podłączenie urządzenia analogowego do cyfrowego, umożliwiającego zapis. Wiele wielofunkcyjnych urządzeń dostępnych na rynku, odtwarzających płyty, ma również wejście na kartę pamięci, a często nawet magnetofon. Jakość takiego sprzętu jest przeciętna i nie mamy praktycznie żadnych zaawansowanych opcji zapisu.

Ograniczenia te sprawiają, że jakość nagrania jest przeciętna, więc więcej sensu ma podłączenie urządzenia do karty dźwiękowej komputera lub adaptera USB, który w wielu przypadkach zawiera też przedwzmacniacz (ramka „Podłączanie”).

Do zapisu potrzebujemy też aplikacji do edycji dźwięku, takiej jak otwartoźródłowe oprogramowanie wykorzystane w tym artykule – Audacity [1], które ma zarówno narzędzia do zapisu, jak i funkcje, dzięki którym poprawimy jakość nagrania.

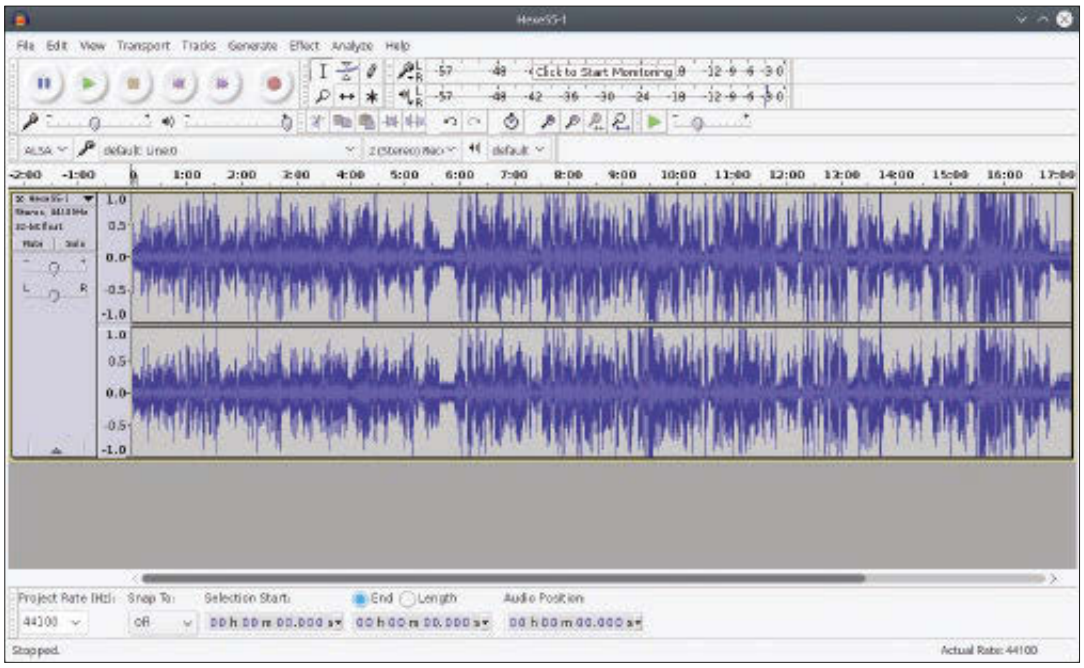
Nagrywanie

Po uruchomieniu Audacity wybieramy najpierw *Plik | Nowy* i zapisujemy projekt. Następnie klikamy na przycisk nagrywania w menu i zaczynamy odtwarzanie kasety lub płyty. W zasadzie możemy teraz spokojnie wypić kawę, podczas gdy oprogramowanie zapisuje muzykę. Warto jednak przyjrzeć się dwóm zielonym wskaźnikom na górze po prawej stronie. Jeśli przesuną się za bardzo w prawo i zmienią kolor na czerwony, powinniśmy albo zredukować siłę sygnału ze wzmacniacza, albo przesunąć suwak w Audacity, a następnie ponowić nagrywanie, aby uniknąć zniekształceń dźwięku.

Najczęściej wystarcza jednokrotne dostosowanie ustawień. Aby odnaleźć największe wychylenia sygnału, warto odsłuchać głośniejsze sekcje i testowo je nagrać. W ten sposób unikniemy zniekształceń i nie będziemy musieli wyrzucać do kosza wykonanego już nagrania całej jednej strony płyty. Po zakończeniu (Rysunek 1) możemy kliknąć na ikony *Powiększ* i *Pomniejsz*, aby obejrzeć szczegółowo cały zapis.

PODŁĄCZANIE

Klasyczny gramofon z igłą, która wędruje po wyżłobieniach płyty, zmieniając je w dźwięki, zwraca sygnał tak słaby, że nie da się go bezpośrednio przesłać na wejście karty dźwiękowej. Chociaż niektóre takie urządzenia posiadają wbudowany przedwzmacniacz, najczęściej będziemy potrzebować wzmacniacza z wejściem phono. Jeśli go nie mamy, możemy rozważyć wykorzystanie przedwzmacniacza USB, szczególnie w przypadku laptopów, które często nie posiadają wejścia audio. Zwykle należy włączyć taką zewnętrzną kartę dźwiękową w opcjach dźwiękowych systemu lub przełączyć się na alternatywne wejście. Magnetofony nie potrzebują wzmacniaczy. Ich wyjście można bezpośrednio podłączyć do wejścia karty dźwiękowej.



Rysunek 1: Po nagraniu oddalamy widok, aby obejrzeć zapis w całości.

Wiosenne porządki

Jako że na płytach mamy zapis analogowy, razem z właściwym sygnałem otrzymujemy też dawkę szumu. Często odtwarzane nagrania zawierają trzaski i zniekształcenia powstałe z powodu zabrudzonych ścieżek. Wyczyszczenie płyty z pomocą szczotki antystatycznej pomaga, ale są też inne sztuczki, dzięki którym wydobędziemy ze starego nośnika czysty dźwięk (ramka „Nowy ze starego").

Wszystkie takie zabiegi nic nie poradzą na zarysowania. Nawet najskuteczniejsze czyszczenie nie powstrzyma więc kliknięć i trzasków. Widoczne są one jako krótkie wahnięcia na ścieżce dźwiękowej. Poziomy wzrastają gwałtownie, co sprawia, że są łatwe do wyodrębnienia. Aby je zlikwidować, wskazujemy myszką daną sekcję i przybliżamy widok za pomocą ikony *Powiększ*, ewentualnie wybierając z menu *Widok | Powiększ* lub korzystając ze skrótu Ctrl+1, aż do chwili, kiedy widzimy pojedynczy impuls (Rysunek 2). Najlepiej działa opcja *Widok | Powiększ zaznaczenie*, dzięki której podświetlony obszar wypełni cały widok, po czym możemy z pomocą myszki zaznaczać kolejny fragment.

Teraz możemy wybrać funkcję *Efekty | Wyciszanie*, która wygładzi impuls. Jednokrotne jej zastosowanie nie sprawi, że dany fragment całkowicie się spłaszczy. Operację musimy powtórzyć kilka razy, dzięki czemu trzaski znikną całkowicie. Pamiętajmy, aby co jakiś czas zapisywać projekt, dzięki czemu nie będziemy ryzykować utraty naszej pracy.

Alternatywą jest półautomatyczne narzędzie do usuwania trzasków zawarte w Audacity. Wystarczy wybrać *Efekty | Usuwanie stukotu* (Rysunek 3). Lekkie zmiany w standardowych ustawieniach dają słyszalne efekty, jednak nie równa się to ręcznemu usuwaniu trzasków, opisanemu wcześniej. Audacity przepuszcza zbyt wiele takich zakłóceń. W skrajnym przypadku funkcja ta potrafi nawet usunąć dźwięki perkusji, więc bądźmy ostrożni, korzystając z automatu.

Ponieważ płyty winylowe zawierają zapis analogowy, automatyczne funkcje nie mają łatwego życia. Żaden algorytm nie usunie typowych dla tego nośnika trzasków. Dzięki wbudowanej funkcji *Efekty | Usuwanie szumu* poprawimy trochę sytuację, ale nie zdziwmy się, jeśli program zapyta nas najpierw o profil szumu: szum może być przeróżny, dlatego Audacity miewa problemy z jego sklasyfikowaniem.

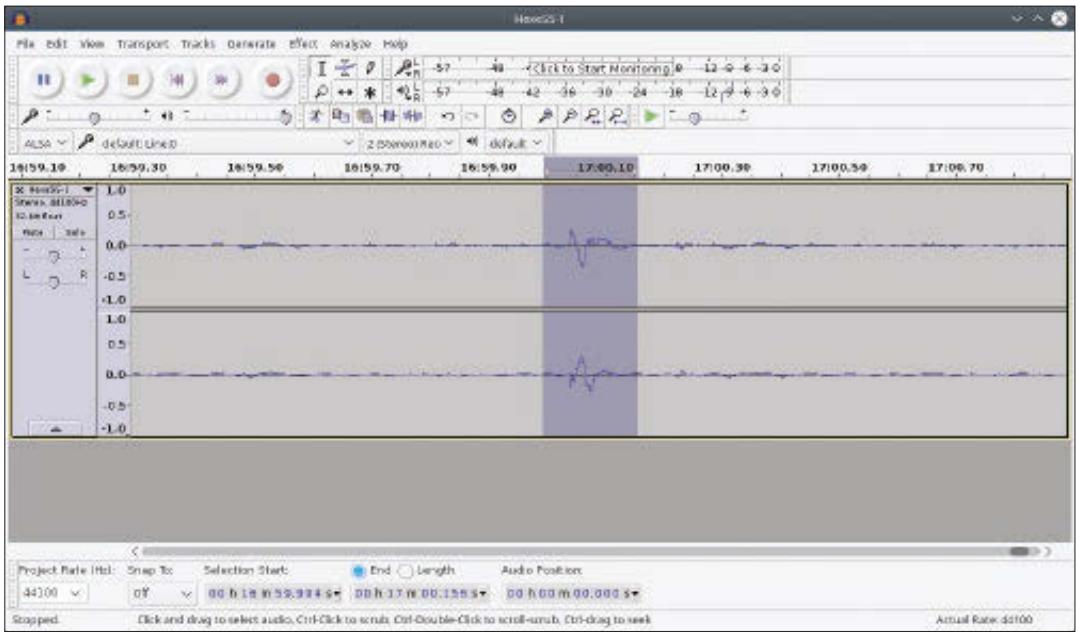
NOWY ZE STAREGO

Oprócz obowiązkowej szczotki mamy jeszcze inne metody usuwania kurzu. Jedną z nich jest wykorzystanie specjalnego ramienia ze szczotką zamiast igły, która dodatkowo rozprowadzi w zabrudzonym rejonie specjalny płyn, dzięki czemu trzaski zostaną w znacznym stopniu zredukowane. Płyn wyparowuje, nie zostawiając żadnych śladów. Jednak płyty odtwarzane, kiedy są jeszcze wilgotne, brzmią, jakby były bardziej zniszczone, niż kiedy są suche.

Alternatywą jest wykorzystanie „zmywarki do płyt", w której dwie szczotki z destylowaną wodą i detergentem czyszczą powierzchnię, a etykieta jest chroniona osłoną. Płytę można odtwarzać dopiero po wyschnięciu.

W przypadku mocno zanieczyszczonych fragmentów płyty, dla których opisane sposoby nie działają, możemy skorzystać ze środka, który dostaniemy w specjalistycznych sklepach. Środkiem tym pokrywa się płytę i ściąga się go po wyschnięciu (jak maseczkę na twarz) razem z cząsteczkami kurzu. Jest on rozpuszczalny w wodzie, więc można go wykorzystać kilka razy. Wystarczy po użyciu rozpuścić i zagotować (chociaż producenci nie polecają takiego rozwiązania).

Rysunek 2: Trzask jest wyraźnie widoczny na powiększeniu.

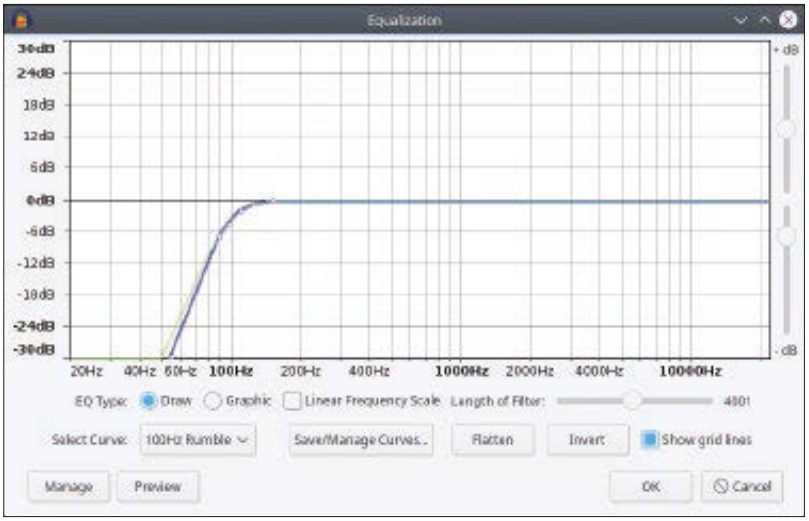


Rysunek 3: Audacity potrafi w półautomatyczny sposób wygładzić trzaski, ale funkcja ta ma swoje ograniczenia.

Zaznaczamy więc mały fragment pomiędzy utworami i klikamy na *Uzyskaj profil szumu*. Program zapisze ten profil w projekcie. Następnie wybieramy *Edycja | Zaznacz | Wszystko*, aby zaznaczyć całe nagranie. W ten sposób usuwanie szumu będzie dotyczyć całości materiału.

Nie oszukujmy się jednak: muzyka nie będzie brzmiała tak świeżo jak zaraz po kupnie płyty. Trzaski są zbyt niejednorodne dla tej metody. Lepsze wyniki uzyskamy, przetwarzając muzykę uzyskaną z kaset, ponieważ tam szum jest tworzony przez powoli przesuwaną taśmę magnetyczną i rozpiętość występujących częstotliwości jest mniejsza.

Rysunek 4: Filtr Rumble jest przydatny, jeśli wzmacniacz nie posiada filtra górnoprzepustowego.



Tworzenie muzyki

Poza usuwaniem szumu Audacity potrafi dokonać cyfrowej korekcji charakterystyki częstotliwości. Typowym przykładem jest filtr eliminujący dudnienie przy niskich częstotliwościach, charakterystyczne dla gramofonów. Po wybraniu *Efekt | Korekcja graficzna*, otrzymujemy okno ustawień (Rysunek 4), w którym możemy wybrać jeden z wielu profilów korekcji. Na przykład *Rumble* to filtr górnoprzepustowy tłumiący niskie częstotliwości. Jeśli ustawione 100 Hz wycina zbyt wiele basów, możemy zmienić krzywą za pomocą myszki.

Zniekształcenie charakterystyki częstotliwości powodowane przez ruch igły na płycie winylowej określone jest w standardzie RIAA (Recording Industry Association of America). Wiele wzmacniaczy (choć nie wszystkie) potrafi to kompensować. Aby usprawnić nagrywanie, powinniśmy przyjrzeć się krzywej korekcji RIAA [2], redukującej wysokie tony i wzmacniającej basy. Czy efekt jest lepszy? To już indywidualna kwestia. Ujednolicone ustawienia częstotliwości niekoniecznie muszą być najlepsze.

Od połowy lat 60. XX w. Dolby Laboratories opracowywało różne metody, z czasem coraz lepsze, pozbycia się szumu taśmy. Szum składa się z bardzo wysokich częstotliwości, więc sygnał w jego obszarze jest zmieniany podczas odtwarzania, w zależności od wejścia, na którym również następuje redukcja szumu. Dolby SR wykorzystywane w studiach potrafi zredukować poziom szumu o 25 dB. Korekcja sprawdza się też w przypadku nagrywania taśm. Magnetofon nagrywający za pomocą technik Dolby [3] zmienia charakterystykę

częstotliwości, co jest uwzględnione, kiedy odtwarzamy taśmę na sprzęcie z technologią Dolby. Jednak Audacity nie posiada takiego profilu. Aby zapoznać się z odpowiednimi standardami, możemy zagłębić się w dyskusję na temat redukcji szumu Dolby prowadzoną w Internecie [4].

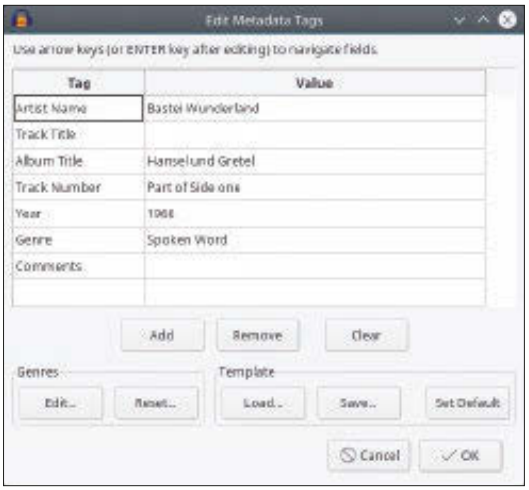
Dziel i rządź

Na płycie, w przeciwieństwie do kasety, możemy gołym okiem zaobserwować przerwy pomiędzy utworami. Jednak kiedy digitalizujemy całość, przerwy znikają, gdyż otrzymujemy jeden duży plik. Teraz musimy podzielić go na mniejsze części.

Nad ścieżką naszego nagrania znajduje się linia czasu, z której teraz skorzystamy. Jeśli w danym albumie są wymienione czasy odgrywania utworów, łatwo możemy obliczyć, gdzie należy przyłożyć wirtualny nóż. Klikamy na początku utworu i przesuwamy zaznaczenie do jego końca. Spadki poziomu sygnału mogą być pewną wskazówką, ale potrafią też zmylić, szczególnie w nagraniach z koncertów.

Okazuje się, że najlepiej jest zacząć od końca nagrania. Łatwiej jest przesłuchać kilka sekund niż cały utwór, tylko po to, aby odnaleźć końcowe jego wyciszenie. Po odnalezieniu końcówki przeciągamy drugi koniec zaznaczenia do miejsca, w którym wydaje nam się, że zaczyna się utwór. Znajdziemy go, odsłuchując krótki fragment nagrania i przesuwając odpowiednio zaznaczenie.

Kiedy zaznaczenie jest już dopasowane, eksportujemy utwór do standardowego formatu akceptowanego przez odtwarzacze. *Plik | Eksportuj zaznaczony dźwięk* otwiera okno dialogowe, w którym domyślnie zapisujemy zaznaczenie do nieskompresowanego formatu WAV. Na dole z prawej możemy wybrać dodatkowe kodeki, takie jak Ogg Vorbis i MP3. Po określeniu formatu pojawiają się dodatkowe opcje, takie jak tryb szybkości transmisji i jakość. Jeśli chcemy skorzystać z egzotycznego kodeka, powinniśmy wybrać *(program zewnętrzny)* zamiast konkretnego formatu, dzięki czemu możemy podać polecenie, jakie ma zostać wykonane w czasie procesu zapisu.



Rysunek 5: Edytor metadanych jest wystarczający, ale odstaje od specjalistycznych narzędzi.

Przetwarzanie albumu od tyłu jest też dobrym sposobem na podzielenie bez żadnej straty nagrań z koncertów. Aby prawidłowo zaznaczyć drugi od końca utwór, przesuwamy po prostu koniec zaznaczenia z ostatniego, do początku przedostatniego utworu. Dzięki temu początek zaznaczenia ostatniej pozycji nagrania staje się końcem dla poprzedniej.

Sekretny kod

Dawniej odtwarzacze uzyskiwały informacje o artyście, albumie i utworze z nazwy pliku. Okładka płyty wyświetlana była, tylko kiedy została zapisana w katalogu z odtwarzanymi plikami. Dzisiaj szczegóły na temat nagrania są zapisywanie jako metadane bezpośrednio w pliku audio. Przy eksporcie do takiego formatu pojawia się odpowiednie okno (Rysunek 5). Podajemy w nim dane takie jak tytuł, artysta, album czy rok wydania.

Domyślnie okno dialogowe posiada jedynie kilka pozycji. Jeśli potrzebujemy więcej, powinniśmy dysponować wiedzą na temat standardów metadanych w danym formacie. Dodatkowo edytor ten nie jest zbyt przydatny, jeśli chcemy uzupełnić informacje o całym albumie. Utwory eksportowane z jednego projektu posiadają te same wpisy co zapisane poprzednio pliki. W wielu przypadkach więcej sensu może mieć eksport nagrań bez metadanych i dodanie ich później, z wykorzystaniem lepszych narzędzi, na przykład EasyTaga [5].

Wnioski

Wkładając niewiele pracy, możemy zachować nasze płyty i kasety w formacie MP3 lub FLAC, aby odżyły w cyfrowym świecie. Program ma pewne wady, na przykład niezbyt praktyczny edytor metadanych, oparty na wxGTK interfejs, który miejscami wygląda brzydko, czy to, że zmiana z Gtk2 na Gtk3 może powodować błędy w wyświetlaniu okienek.

Nie ma jednak zbyt wielu alternatyw. EKO [6] i podobne, proste edytory dźwięku są dosyć ograniczone. Tylko KWave [7] oferuje funkcjonalność podobną do Audacity. Poza wymienionymi aplikacjami trudno coś znaleźć w repozytoriach.

INFO

[1] Audacity: http://www.audacityteam.org
[2] Korekcja RIAA: https://en.wikipedia.org/wiki/RIAA_equalization
[3] Systemy redukcji szumu: https://en.wikipedia.org/wiki/Dolby_Laboratories
[4] Standardy Dolby: http://hyperphysics.phy-astr.gsu.edu/hbase/Audio/tape5.html
[5] EasyTag: https://wiki.gnome.org/Apps/EasyTAG
[6] EKO: http://semiletov.org/eko/
[7] KWave: https://www.kde.org/applications/multimedia/kwave/

Perełki FOSS

NAJCIEKAWSZE PROJEKTY ZE ŚWIATA WOLNEGO I OTWARTEGO OPROGRAMOWANIA

Graham Morrison przerywa aktualizowanie Arch Linuksa, by ruszyć na poszukiwanie najlepszych nowinek w świecie wolnego oprogramowania.

Minimalistyczna przeglądarka

qutebrowser 1.0

Od kiedy jakiś czas temu zacząłem używać Qutebrowsera, zmienił on sposób, w jaki myślę o Linuksie i projektowaniu interfejsu użytkownika. Minimalizm i podejście oparte na dobrze znanych skrótach klawiszowych Vima oszczędzają czas i zmniejszają ilość problemów. Przeglądanie Internetu jest szybsze, nic nie rozprasza uwagi, a całość jest intuicyjna. Nawet jeśli pamiętamy jedynie kilka poleceń Vima, wystarczy to do otwarcia nowej karty, wybrania adresu, przypisania podpowiedzi do linków na stronie i zapisania zakładek. Istnieją dodatki do Chromium i Firefoksa, które realizują takie same zadania, ale Qutebrowser bije je na głowę stopniem integracji z przeglądarką i rozmiarem całej aplikacji. Jeśli jeszcze jej nie używaliście, wersja 1.0 to najlepsza wymówka, aby przypomnieć sobie umiejętności pracy z Vimem.

Po dwóch udanych zbiórkach funduszy na rozwój projektu Qutebrowser 1.0 realizuje wszystko, co planowano, w szczególności zmiany w opcjach konfiguracji programu. Niestety oznacza to, że nie możemy automatycznie migrować ustawień podczas aktualizacji ze starszej wersji. Ma to duże znaczenie, ponieważ jedną z lepszych cech Qutebrowsera jest olbrzymia wszechstronność konfiguracji, dzięki poleceniom *:set* oraz *:save*. Szczęśliwie zakładki i historia są przenoszone ze starszej wersji, a nowy widok *diff* pozwala sprawdzić różnice pomiędzy konfiguracją w wersji 1.0 a wcześniejszą. Dzięki temu dosyć łatwo jest skopiować nasze stare ustawienia do nowego formatu, a plik z konfiguracją jest dużo łatwiejszy do edycji i utworzenia niż poprzednio.

1. DuckDuckGo: wciskamy *o* i wpisujemy szukaną frazę. Wynik pojawia się natychmiast. Dodajemy *!g*, aby wyszukać przez Google. **2. Karty:** karty można zapisać, przemieścić i przypiąć. **3. Podpowiedzi:** dzięki skrótom, nawigacja bez myszki jest szybka i prosta. **4. Różnice w konfiguracji:** przechodzimy do *qute://configdiff*, aby sprawdzić różnice w stosunku do starej konfiguracji. **5. Nowa konfiguracja:** opcje konfiguracji w tej wersji zostały gruntownie przerobione. **6. Uzupełnianie poleceń:** tak jak w Vimie, posiłkujemy się pomocą i uzupełnianiem poleceń. **7. Zapis:** automatycznie lub ręcznie zapisujemy i odzyskujemy naszą sesję.

Inną ważną zmianą w wersji 1.0, poza dziesiątkami małych poprawek, dzięki którym program jest stabilniejszy, jest to, że QtWebEngine staje się domyślnym silnikiem renderującym. Oznacza to, że strony takie jak Facebook, GitHub, Gmail, TweetDeck i Dokumenty Google będą działać bez żadnych problemów. Ilość niekompatybilnych witryn gwałtownie zmalała. Razem z nowym silnikiem dołączono sprawdzanie pisowni, chociaż najpierw należy uruchomić skrypt Pythona, który pobierze wymagane słowniki, a następnie wydać polecenie konfigurujące. Dzięki temu błędy w pisowni będą podświetlane, tak jak w Chromium, co ma duże znaczenie dla mających problemy z pisownią, gdy dodają nowe wpisy na Tweeterze czy też piszą e-maile. Czasami wręcz nie zdajemy sobie sprawy z tego, że polegamy na tej opcji dopóty, dopóki jej nie zabraknie. Poza tymi zmianami dostępnych jest wiele nowych opcji konfiguracji i całkowicie zaktualizowana historia, w której przechowywane są teraz wszystkie odwiedzane strony lub, opcjonalnie, tylko ich konkretna ilość. To jest właśnie piękne w Quteborwserze. Nie tylko korzysta on (domyślnie) ze skrótów Vima i nie obciąża zanadto systemu, ale może też być skonfigurowany tak, że spełniać będzie każdą funkcję, jakiej oczekujemy od przeglądarki. Jedynym wyjątkiem jest interfejs wtyczek, co – miejmy nadzieję – zostanie dodane w kolejnej dużej aktualizacji. ■■■

Strona projektu

https://www.qutebrowser.org/

Minimalistyczna przeglądarka

Min

Usuwamy nadmiarowe obiekty z przeglądanych stron z pomocą potężnej, szybkiej i minimalistycznej przeglądarki.

Miejmy nadzieję, że zaczyna się nowa era przeglądania Internetu. Taka, w której treść, prywatność i wygoda nawigacji są ważniejsze niż wszędobylskie reklamy, wyskakujące okienka, prośby o udostępnienie lokalizacji i automatycznie odtwarzane filmiki. Pierwsze jaskółki zwiastujące nową jakość to proste, nierozpraszające przeglądarki taki jak Qutebrowser (omawiany wcześniej), wspaniały Firefox Focus na Androida i nowa aplikacja tego typu Min. Min zasługuje na swoją nazwę od pierwszego uruchomienia, kiedy pyta, czy chcemy blokować reklamy, skrypty i obrazy oraz śledzić zachowania użytkownika. Możemy też aktywować bardzo przydatny tryb nocny (dark mode), którego nie oferują nawet przeglądarki głównego nurtu.

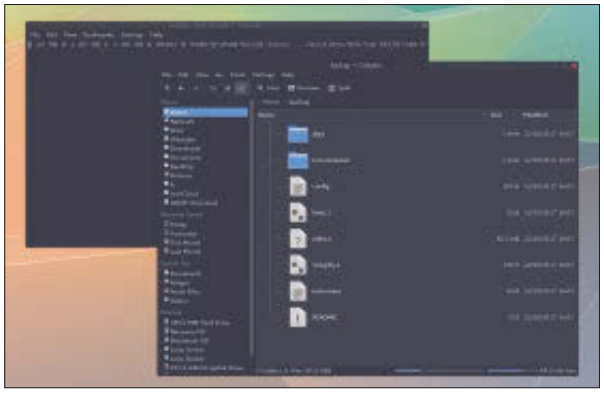
Przeglądanie też jest dopracowane, a interfejs użytkownika skupia się na wynikach wyszukiwania. Zaczynamy, wprowadzając w dowolnym miejscu tekst i otrzymujemy w czasie rzeczywistym sugestie, które pojawiają się zwykle na górze głównego okna. Są one pobierane domyślnie z DuckDuckGo, wyszukiwarki, która wreszcie zaczyna się odnajdywać w swej roli. Możemy otworzyć wiele kart, które grupujemy w „zadania" będące wspaniałym sposobem na oddzielenie pracy i niezwiązanych z nią stron. Karty możemy też wyświetlić w formie listy. Istnieje również tryb eliminujący rozpraszacze, dzięki któremu skupimy się na jednej karcie i który nie pozwoli nam otworzyć nowych, co jest przydatne, jeśli nie chcemy, aby kusił nas Reddit. Dodatkowo, istnieje mnóstwo skrótów klawiszowych, a nawet gestów ułatwiających nawigację. Jedyną wadą jest to, że przeglądarkę napisano z wykorzystaniem Electrona z Javasciptem i CSS, ale jej szybkość nie zdradza tego pochodzenia, a interfejs przyciąga wzrok. ■■■

Strona projektu

https://minbrowser.github.io/min/

Kopia zapasowa

BorgBackup 1.1



Interfejs wiersza poleceń jest trochę surowy, możemy go więc zastąpić stroną internetową w roli interfejsu graficznego.

BorgBackup to narzędzie do wykonywania kopii zapasowych z poziomu wiersza poleceń, ale ma kilka cech, dzięki którym jest lepszym wyborem niż na przykład *tar cvzf*. Twórcy mówią o nim „nieduplikujący program do kopii zapasowej". Oznacza to, że jego celem jest przechowywanie jedynie zmian w plikach, zamiast wielu kompletnych duplikatów, jak w *rsyns* kopiującym katalog do nowej lokalizacji. Z samego założenia jest to sposób oszczędzający miejsce oraz transfer i wiele programów korzysta z tego mechanizmu. BorgBackup realizuje to zadanie, dzieląc plik na kawałki i generując dla każdego sumę kontrolną. Następnie kopiuje tylko te kawałki, których sumy kontrolnej jeszcze nie widział. Po połączeniu z szyfrowaniem po stronie klienta oraz z kompresją BorgBackup staje się doskonałym narzędziem, wyciągającym nas ze stanu apatii, której objawem jest zdanie: „nie ma szans, żeby to się wydarzyło".

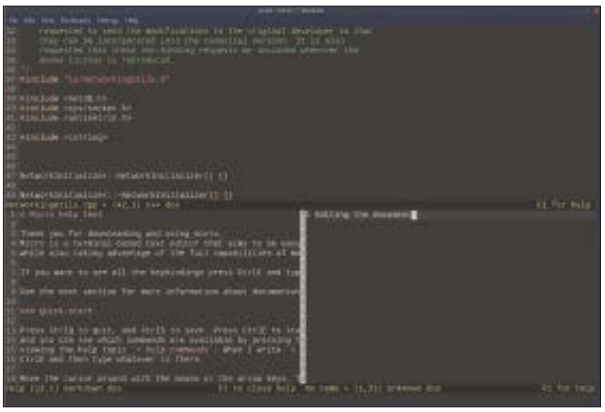
W przeciwieństwie do *rsynca*, BorgBackup jest prostszy w użyciu. Najpierw, za pomocą polecenia *init*, inicjujemy repozytorium kopii, analogicznie do polecenia *git*, a następnie z pomocą *create* tworzymy zadanie kopiujące, korzystające z informacji o źródle i docelowym katalogu. Kopia tworzona jest bez wyświetlania w terminalu postępów, chyba że dodamy opcje *--stats* i *--progress*. Dostępnych jest wiele innych opcji, z których możemy skorzystać, aby dostosować lepiej cały proces kopiowania, jak też odzyskiwania plików. Aby utworzyć kolejną kopię, wykonujemy to samo polecenie, dzięki czemu automatycznie zrealizujemy kopię przyrostową, zapisując jedynie zmiany w repozytorium. Działa to doskonale, a ostatnio program dostał dużą aktualizację. Ponad 60 współtwórców pomogło dodać nowe funkcje, dzięki którym otrzymujemy możliwości usuwania plików z istniejącego archiwum, automatyczną kompresję, większą ilość metod szyfrowania, lepszy dziennik i mnóstwo ulepszeń zwiększających prędkość działania i bezpieczeństwo aplikacji. ■■■

Strona projektu

https://www.borgbackup.org/

Minimalistyczny edytor tekstu

Micro



Doskonały, podzielony ekran oraz podświetlanie składni w programie zajmującym mniej niż 10MB.

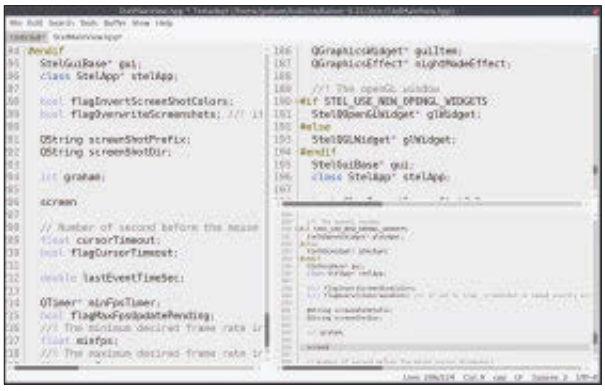
Pozostajemy w temacie minimalistycznych rozwiązań i przedstawiamy Micro, który dla edycji tekstów jest tym, co Min dla przeglądania sieci. Już uruchomienie go z wiersza poleceń pokazuje, z czym mamy do czynienia. Działa w terminalu, tak jak Vim czy Nano, a nie Gedit i Kate. Dodatkowo jest to binarny pakiet statyczny, więc bez problemu możemy go przechowywać na pamięci USB czy jako załącznik do e-maila i uruchamiać pojedynczy plik (jeśli ufamy temu wydaniu) zawsze wtedy, kiedy chcemy edytować tekst. Podobnie jak w Vimie i w przeciwieństwie do Nano, nie od razu możemy zorientować się, jak wykonywać proste operacje, takie jak na przykład wyjście z programu, ale łatwo jest to odkryć: wystarczy wcisnąć F1, aby otworzyć pomoc. Zamknięcie aplikacji opisanie jest w pierwszym paragrafie (Ctrl+q). Dokumentacja jest krótka, ale wyczerpująca i zawiera nawet samouczek.

Jednak zredukowany do minimum interfejs nie oznacza minimalnej funkcjonalności. Podział ekranu jest łatwo dostępny w trybie wpisywania samouzupełniających się poleceń. Micro wspiera ponad 75 języków. Podświetlanie ich składni pojawia się bez opóźnień i automatycznie, w zależności od typu pliku, który edytujemy. Chociaż szybkość pisania i edycji to wartości bardzo subiektywne, Micro wydaje się bardzo szybki. Ponieważ przypisanie klawiszy łatwo zmienić, możemy ustawić skróty, do których jesteśmy przyzwyczajeni. Dzięki pisanym w LUA wtyczkom rozszerzymy aplikację, dodając między innymi definiowanie powtarzalnych fragmentów kodu (snippets), sprawdzanie pisowni, konfigurację edycji inline i kontrolę wersji. Napisanie własnej wtyczki również nie stwarza problemów. Micro nie zastąpi Vima czy Emacsa, jest jednak bardzo dobrym dodatkiem, przydatnym, kiedy potrzebujemy małego, szybkiego i funkcjonalnego edytora. Jedynym, który może się z nim równać w tej kategorii wagowej, jest Textadept. ■■■

Strona projektu
https://micro-editor.github.io

Minimalistyczny edytor tekstu

Textadept



Jak wiele edytorów zajmuje mniej niż 15 MB i ma zmienne rozmiary czcionek oraz nieskończoną możliwość pionowego i poziomego podziału ekranu?

Edytory tekstu to jeden z liczniejszych rodzajów oprogramowania. Oczywiście istnieją stare rozwiązania, które ciągle wywołują wiele dyskusji, ale ciągle też pojawiają się nowi członkowie tej rodziny, a każdy z nich inaczej podchodzi do wstawiania litery za literą. Textadept, tak jak Min i Micro, stawia na minimalizm, obiecując przy tym prędkość oraz nierozpraszający interfejs, bez poświęcania funkcjonalności. Szczęśliwie, skupia się na czymś innym niż Min, gdyż skierowany jest do programistów. Nie wspiera zbyt wielu języków. Aktualnie jest ich około 100, dla których potrafi podświetlać składnię, ale duża liczba nie ma tu znaczenia, gdyż 90 procent kodu powstaje w jedynie kilku językach. Poza tym w Textadepcie edycja kodu jest lepiej rozwiązana niż w Minie.

Textadept to dojrzałe rozwiązanie. Ma już ponad dziesięć lat, z czego od sześciu, co dwa miesiące, wydawana jest nowa wersja. Edytor można uruchomić z wiersza poleceń (gdzie posiada interfejs napisany z wykorzystaniem curses) lub w osobnym oknie. Plik wykonywalny zajmuje około 5 MB, można go uruchomić z pamięci USB, przy czym alokuje nie więcej niż 15 MB RAM-u. W tak niewielkiej przestrzeni zmieściło się nie tylko niezwykle szybkie podświetlanie składni, ale też co ważniejsze, uzupełnianie kodu. Dla tych z nas, którzy nie mają fotograficznej pamięci lub odeszli od przepisywania kodu z żółtych stron magazynów, jest to niezbędna opcja. Autouzupełnianie działa z symbolami w edytowanych plikach, jak też symbolami w języku, w którym pracujemy, dostarczając jednocześnie linków do dokumentacji. Edytor można obsługiwać jedynie klawiaturą, jest łatwy do opanowania i dobrze udokumentowany. ■■■

Strona projektu
https://foicica.com/textadept/

Klient poczty

Geary 0.12

Dzięki temu, że wygląda zgrabnie i mądrze, grupuje wymiany e-maili, Geary ułatwia odejście od klienta poczty w przeglądarce.

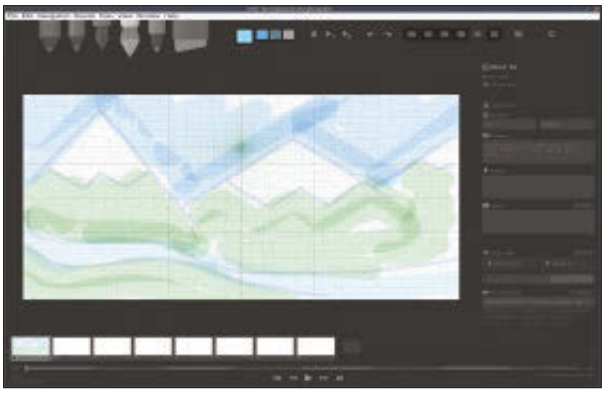
Programy pulpitu do odbioru i zarządzania pocztą elektroniczną nie powinny odchodzić w zapomnienie, pomimo że wielu użytkowników korzysta na co dzień z klientów webowych. Są niezbędne, jeśli na przykład chcemy wykonywać kopię zapasową poczty, co jest koniecznością. Spełniają też pożyteczną funkcję, będąc narzędziami utrzymującymi nas z dala od pokus przeglądarki. Problemem jest to, że w tej dziedzinie popularność aplikacji webowych sprawia, iż wersje pulpitowe wymierają. Większość popularnych klientów pod Linuksem idzie tą samą drogą co niesamowita Eudora. Geary, który przez miesiące pozostawał w stanie zawieszenia, udowadnia ostatnią aktualizacją, wydaną prawie 18 miesięcy po wersji 0.11, że jest wyjątkiem. Stał się teraz fragmentem projektu Gnome, na co tak doskonały program w pełni zasługuje.

Najlepsze w Gearym jest to, że korzysta z Vala/GTK+, dzięki czemu dysponuje możliwościami dostępnymi w Gnome'ie. Jego zgrabny projekt jest lepszy niż odpowiedników webowych. Wyświetla więcej informacji i jest bardziej wydajny. Przejścia, dla przykładu, są przepiękne, a sposób, w jaki górna belka jest wykorzystywana jako pasek narzędzi i menu, sprawia, że całość wygląda czysto nawet w środowisku KDE, z całkowicie innym menedżerem okien. Podobnie jak w Gmailu, z którym integracja nie sprawia problemów, słowa kluczowe mogą być wykorzystane do grupowania i przeszukiwania skrzynki odbiorczej. Wymiany korespondencji grupowane są w pionie, a Geary jest jedyną aplikacją pulpitu pod Linuksem, która robi to dobrze. Nowa aktualizacja ulepsza stabilność i usprawnia redagowanie wiadomości w różnych stylach. Najlepsze jest jednak to, że możemy, wciskając Ctrl + ? (Shift+/), zobaczyć stronę podpowiadającą skróty do trybu, w którym aktualnie się znajdujemy, sprawiając, że nauka poruszania się po aplikacji jest bezproblemowa. ■■■

Strona projektu
https://github.com/GNOME/geary

Planowanie scenariusza

Storyboarder



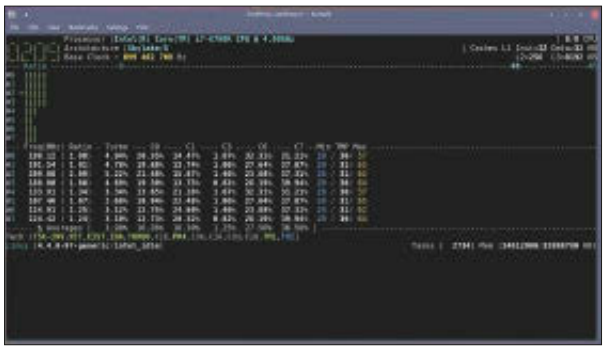
Zaplanujmy każde ujęcie naszego kolejnego, genialnego dzieła na YouTubie, z pomocą wspaniałego (prawie) otwartego oprogramowania.

Po pierwsze, pomimo że projekt określa się jako „otwarty", jego otwartoźródłowość nie do końca spełnia przyjętą definicję. Jest na licencji MIT z dodatkiem kilku wyjątków, które uniemożliwiają zarabianie na kodzie. Wyjątki te spowodowane są doświadczeniami dewelopera z ludźmi dokonującymi niewielkich zmian w projekcie i usuwającymi odniesienia do autorów oprogramowania. Dlatego właśnie musimy podać adres poczty internetowej, jeśli chcemy pobrać aplikację, mimo że kod umieszczony jest na GitHubie. Projekt wart jest jednak uwagi, ponieważ jest to doskonały sposób, szczególnie dla dzieci, na eksperymentowanie z planowaniem i tworzeniem historii. Pod Linuksem nie ma zbyt wiele takich aplikacji. Ponieważ ciągle jest to świeże oprogramowanie, możliwe, że po pewnej dozie dyskusji i przy odpowiednim wsparciu deweloper może w przyszłości zmienić zdanie co do przejścia na prawdziwie otwartą licencję.

Tworzenie scenorysu polega na rozrysowaniu planu historii na kartach, tworząc w ten sposób podstawowy, rysunkowy scenariusz, który może zostać wykorzystany przez reżysera przy planowaniu kolejnych ujęć, kompozycji planu i poruszania się jego elementów. Storyboarder pomaga nam w tym zadaniu, udostępniając różne narzędzia do rysowania, tak jak Krita lub Gimp, oraz dodatkowe pola, w którym możemy umieścić opis ujęcia czy dialog. Program stara się uprościć proces, dając do naszej dyspozycji ograniczoną paletę i małą ilość narzędzi. Z jego pomocą nie powstanie plan, w którym każda klatka to arcydzieło, nie do tego on służy. Mamy przelać nasze pomysły na karty scenorysu, obrazując naszą wizję. Możemy „odegrać" wszystkie karty, aby sprawdzić, jak wygląda opowieść, zmieniać poszczególne sceny i ich kolejność. Aplikacja sprawia profesjonalnie wrażenie, mimo że proponuje edycję kolejnych scen w Photoshopie. Jest to doskonały sposób na zaplanowanie krótkiego filmu czy animacji, szczególnie jeśli prowadzimy zajęcia o filmie lub uczymy dzieci, jak zaplanować swój projekt. ■■■

Strona projektu
https://github.com/wonderunit/storyboarder

Monitorowanie procesora
CoreFreq



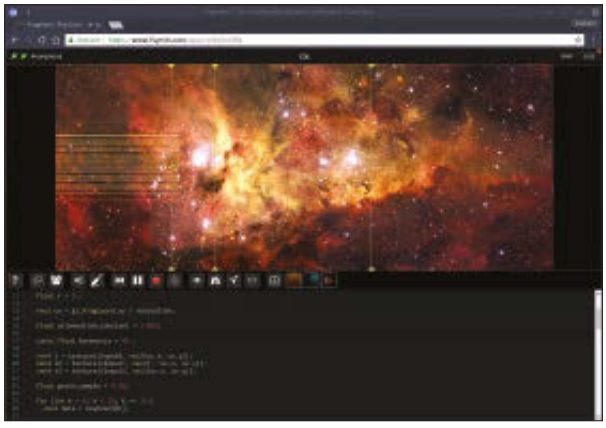
CoreFreq wyświetla niespotykaną ilość informacji na temat naszego procesora, jak też dane na temat wykorzystania go przez aplikacje i system operacyjny.

Nie bylibyśmy sobą, gdybyśmy nie opisali w tym numerze chociaż jednego narzędzia monitorującego wydajność. Tym razem trafiliśmy na całkiem niezłe. CoreFreq podaje informacje o procesorze i skierowany jest raczej do nowszego sprzętu. Działa pod 64-bitowym systemem, razem z procesorami takimi jak Intel Atom, Core 2, Nehalem, Sandy Bridge lub lepszymi. Użytkownicy AMD powinni dysponować procesorem z rodziny 0Fh (AMD K8 Hammer) lub czymś nowszym. Powodem jest to, że program obiecuje wysoki poziom precyzji i został napisany do monitorowania współczesnych technologii wykorzystywanych w procesorach (SpeedStep – EIST, Turbo Boost, Hyper-Threading – HTT i Base Clock). CoreFreq dostarcza także informacji na temat oprogramowania korzystającego z procesora, wliczając w to ilość instrukcji na cykl lub sekundę, IPS, IPC lub CPI, stan C, temperaturę, jak również informacje z różnego rodzaju liczników wydajności.

Taka ingerencja w działanie procesora nie odbywa się bez kosztu. Jest nim ilość uprawnień, jakich wymaga CoreFreq. Nie tylko potrzebuje własnego demona z uprawnieniami roota, ale też osobnego modułu jądra. Jest to zrozumiałe, jeśli bierzemy pod uwagę, jak działa monitorowanie, ale warto to przemyśleć, jeśli uruchamiamy program na newralgicznej maszynie. Po zainstalowaniu i uruchomieniu domyślny widok programu przypomina monitor Htop. Każdy rdzeń dostaje własny histogram oraz tablicę, w której znajdują się statusy wspomniane wcześniej. Z pomocą menu możemy przełączać się pomiędzy trybami i wyświetlać różne parametry. Dostępny jest też prosty menedżer wyświetlania, dzięki któremu wyświetlimy nad monitorem szczegóły, takie jak topologia procesora i informacje o sprzęcie. Dodatkowe dane można sprawdzić z pomocą opcji w wierszu poleceń, dzięki czemu CoreFreq jest jednym z najbardziej wszechstronnych programów do monitorowania procesora. ■■■

Strona projektu
https://github.com/cyring/CoreFreq

Granularny syntezator dźwięku
Fragment



Korzystamy z mocy procesora graficznego, aby wytworzyć dziwne i wspaniałe dźwięki.

W tym miesiącu kategorię audio reprezentuje Fragment, czyli „wieloplatformowe środowisko współpracy przy tworzeniu utworów audiowizualnych, z nowym podejściem do syntezy dźwięku opartej na synchronizacji z obrazami z dziedziny piksel artu”. Jeśli chcemy zrozumieć to zdanie, musimy rozbić je na mniejsze części i przyjrzeć się kolejnym słowom. Jednak już samo skomplikowanie tego zdania oddaje złożoność i możliwości oprogramowania. W skrócie, Fragment tworzy dźwięki, ale są one raczej programowane niż odgrywane. Zadania możemy realizować razem z innymi ludźmi, w czasie rzeczywistym, edytując treść podczas odtwarzania. Muzyka generowana jest z informacji zawartych w pikselach, wytworzonych przez nasz kod zgodny z OpenGL Shading Language (GLSL). Jest to utwór audiowizualny, ponieważ wejście może być obrazem, filmem i muzyką, czyli shaderami w GLSL. Jest to podejście granularne, ponieważ wyjście generowane jest poprzez pobieranie małych próbek ze źródeł, przetwarzanie ich i zwracanie w postaci chmury pomieszanych razem dźwięków. Fragment jest tak skomplikowany jak ten opis i trudny do zrozumienia, tym bardziej że dokumentacja i przykłady są dość ubogie. Jednak jego brzmienie jest wprost wspaniałe.

Fragment jest dość niecodzienny: chociaż działa lokalnie, stworzono go z wykorzystaniem technologii webowych i jest dostępny przez przeglądarki takie jak Chromium i Firefox. Uruchamiamy kod i wysyłamy nuty do syntezatora, wykorzystując albo OSC, albo MIDI, chociaż Firefox nie wspiera aktualnie tego drugiego formatu. Edycji możemy dokonywać w czasie rzeczywistym. To jest ta część, w której możliwa jest współpraca. Oprogramowanie pozwala realizować swoje pomysły przez sieć, bez konieczności instalacji. Ładujemy obrazy, definiujemy je w kodzie jako wejście i rozpoczynamy przetwarzanie. ■■■

Strona projektu
https://www.fsynth.com/

Kolekcjonerska gra karciana

Argentum Age

Nie ma zbyt wielu gier karcianych, które by miały tak piękną i dojrzałą szatę graficzną.

Wydawać by się mogło, że gry w karty na komputerze nie mają przyszłości, jednak tego typu rozrywka zyskuje na popularności. Ciągle jednak jest niewiele otwartoźródłowych gier tego typu. Dzieje się tak, ponieważ zwykle opierają się na wspaniałej grafice, za którą stoją artyści, niebędący zwykle, tak jak deweloperzy, przekonani do idei otwartych źródeł. Grafiki w Argentum Age są (niemal) wyjątkiem. Wydane na mieszance licencji CC BY-NC-ND, GPL i CC0, są bardziej otwarte niż grafiki w wielu podobnych grach, a dodatkowo treść, jak też licencje mogą się zmienić.

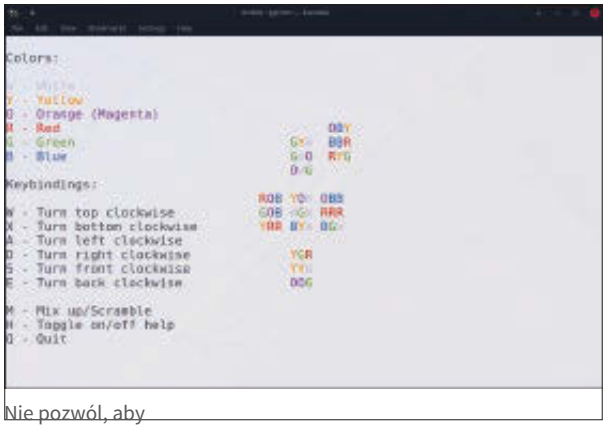
Kolekcjonerskie gry karciane (CCG) to specyficzny gatunek, w którym gracz zbiera karty i tworzy własne talie. Każda z nich opiera się na różnych umiejętnościach i cechach opisanych na kartach. Gracze rywalizują z ich pomocą, na zasadach analogicznych do gier planszowych. Jeden typ kart może być wykorzystany do przyzwania istot ofensywnych, podczas gdy inny może rozszerzyć ilość kart możliwych do posiadania na ręce. Większość tego typu gier działa na podobnych zasadach, a różni je historia i świat przedstawiony oraz konkretne reguły specyficzne dla danej produkcji. W Argentum Age na przykład liczba kart na ręce odpowiada wielkości wioski. Zrozumienie zasad i rozpoczęcie gry może być wyzwaniem, ale satysfakcja ze skompletowania talii i umiejętnego nią operowania jest olbrzymia. Argentum Age zawiera tryb dla pojedynczego gracza, więc możemy eksperymentować z kartami, obserwując rozwijającą się historię. Możemy też sprawdzić się w sieci, weryfikując nasz dobór kart w starciu z żywymi ludźmi. W grze dostępny jest czat oraz wsparcie społeczności. ■■■

Strona projektu

http://argentumage.com/

Kostka w wierszu poleceń

NRubik



Nie pozwól, aby podczas pracy w biurze klikanie fizycznej kostki powstrzymało Cię od rozwiązywania tej nieśmiertelnej zagadki.

Kostka Rubika, jej różnorakie kopie i wariacje ciągle są popularne. Dzieje się tak, dlatego że jest to prawdopodobnie jedna z najciekawszych zabawek, jakie kiedykolwiek powstały. Poruszanie nią sprawia dziwną przyjemność, łatwo zacząć zabawę i trudno jest ułożyć kostkę. Nawet jeśli opanowaliśmy podstawy rozwiązywania tej zagadki, szybko ogarnie nas obsesja skrócenia czasu układania. Zwykle zaczynamy od pięciu minut, planując skrócić ten czas do mniej niż 30 sekund. Wszystkie istniejące rozwiązania wymagają dobrej pamięci, zręczności i wytrwałości. Możliwe, że dlatego kostka jest ciągle popularna. Kiedy świat na tak wiele sposobów rozprasza naszą uwagę, skupienie się na kostce na parę minut to wspaniały sposób na odprężenie. Chociaż fizyczna wersja zawsze będzie tą najlepszą, wirtualne odpowiedniki oferują własne wyzwania, zmuszając do przełożenia naszej wiedzy do świata 2D. Jest to dosyć duże wyzwanie.

Na pulpicie istnieje kilka dobrych aplikacji emulujących kostkę Rubika, od projektów 2D, po oparte na OpenGL-u wersje 3D. Najłatwiejsze w użyciu są te, w których możemy z pomocą myszki emulować ruchy rąk. Problemem jest to, że najczęściej nie możemy w łatwy sposób szybko zobaczyć, co jest po drugiej stronie. Nie jest to problemem w przypadku aplikacji NRubik, ponieważ uruchamiamy ją z wiersza poleceń, a ona sama wykorzystuje bibliotekę Curses do przedstawienia prostego widoku wirtualnej kostki. Wszystkie możliwości kontrolowania jej wyświetlone są na ekranie. Pozwalają na obrócenie dowolnego rzędu, jak też na pomieszanie całości. Podane klawisze w pełni wystarczają do znalezienia rozwiązania. Chociaż jest to trudne, nawet jeśli pamiętamy sekwencję do niego prowadzącą. Wyzwanie jest prawdziwe i to ono sprawia, że popularność kostki utrzymuje się tak długo. ■■■

Strona projektu

https://github.com/cheertarts/nrubik

Zaawansowana edycja filmów za pomocą FFmpega

Filmowy czarodziej

W Linuksie istnieje wiele znakomitych narzędzi do edycji filmów działających w trybie graficznym, czasem jednak lepiej jest użyć wiersza poleceń i narzędzia FFmpeg. Paul Brown

Kiedy warto użyć FFmpega? Przede wszystkim wtedy, gdy musimy przetwarzać pliki wsadowo. Załóżmy, że mamy zamienić w 100-stronicowym dokumencie wszystkie wystąpienia słowa „Jan" na „Kazimierz". Możemy oczywiście zrobić to ręcznie: otworzyć plik w edytorze tekstu i użyć funkcji „Wyszukaj i zamień". Jeśli zmiany mamy wprowadzić w jednym pliku, jest to dobre rozwiązanie, co jednak w sytuacji, kiedy przetworzyć trzeba setki dokumentów porozrzucanych po systemie plików? Raczej nie rozważalibyśmy ręcznego przeszukiwania katalogów i otwierania każdego pliku oddzielnie, prawda? Znacznie lepszym rozwiązaniem byłby skrypt powłoki z Findem i Sedem.

Możemy wierzyć lub nie, ale to samo dotyczy edycji filmów. Istnieją dziesiątki, a może i setki rzeczy, które możemy zrobić z filmami bez konieczności dotykania interfejsu graficznego: wystarczy nam FFmpeg [1].

Prawdopodobnie używaliśmy już kiedyś tego narzędzia do konwersji plików w różnych formatach audio i wideo, np. poniższe polecenie przeprowadzi konwersję pliku MP4 do formatu WebM:

```
ffmpeg -i input.mp4 output.webm
```

FFmpeg potrafi jednak znacznie więcej. Może modyfikować liczbę klatek na sekundę, przełączać ścieżki dźwiękowe i napisów, a nawet przycinać i zamieniać miejscami poszczególne fragmenty filmu.

Rysunek 1: Logo służące do oznaczenia filmu znakiem wodnym.

Umieszczamy znak wodny.

Jednak jedną z najpotężniejszych funkcji FFmpega jest zestaw filtrów [2]. Możemy je nałożyć albo na całe ścieżki z dźwiękiem i obrazem, albo na niektóre fragmenty, uzyskując interesujące efekty. Aby zilustrować działanie filtrów, pokażemy, w jaki sposób użyć logo w roli znaku wodnego nałożonego na film. Przykładowe logo widzimy na Rysunku 1: jest to plik NG z przezroczystym tłem. Zakładamy, że pracujemy na pliku MP4 720p (1280 × 720) o nazwie *film.mp4*.

Wspomniany efekt możemy osiągnąć na wiele sposobów, skupimy się jednak na najprostszym z nich, wykorzystującym filtr *overlay*.

Polecenie z Listingu 1 (które można zmieścić w jednym wierszu) przyjmuje dwa pliki wejściowe: plik wideo *film*.mp4 oraz obraz *LM_logo.png*, po czym wyświetla je razem – drugi nałożony na pierwszy, w pliku *film_oznaczony.mp4*. Wynik działania polecenia widzimy na Rysunku 2.

Na uwagę zasługuje konstrukcja *-filter_complex* w wierszach 2–4, która znajduje się między wejściami a wyjściem. W ramach *-filter_complex* możemy łączyć różne filtry, a zostaną one nałożone po kolei na jeden strumień, drugi bądź obydwa.

Choć uczyniliśmy krok we właściwym kierunku, rezultat nie wygląda zbyt subtelnie. Logo jest w niewłaściwym miejscu: zamiast w lewym górnym, powinno być w prawym dolnym rogu.

Na szczęście większość filtrów przyjmuje parametry, a *overlay* nie jest wyjątkiem (Listing 2).

Kiedy przekazujemy parametr filtrowi, używamy składni *filtr=wartość*. W tym przypadku przekazujemy *overlay* umiejscowienie w pionie i poziomie, przy czym obie te wartości oddzielone są dwukropkiem.

FFmpeg pozwala też w wygodny sposób przekazać szerokość i wysokość filtrowi nakładki wysokość i szerokość każdej warstwy: *W* to szerokość pierwszego wejścia (warstwy dolnej), natomiast *w* to szerokość drugiego wejścia (warstwy górnej). Oznacza to, że *W-w-10* nałoży górną warstwę 10

pikseli od lewej krawędzi dolnej warstwy z filmem. To samo dotyczy *H-h-10*, tyle że odnosi się do osi pionowej (Rysunek 3).

Logo jest jednak nadal zbyt duże. Możemy rozwiązać ten problem, dodając nowy filtr i doczepiając go do *overlay* (Listing 3).

Pojawiło się tu kilka nowych elementów. Przede wszystkim widzimy nowy filtr *scale*, który zmienia skalę strumienia wideo, przyjmując jako argumenty nową szerokość i wysokość oddzielone dwukropkiem. Jeśli chcemy zapewnić utrzymanie proporcji, należy przekazać tylko jeden z parametrów, zastępując drugi stałą *-1*. W przykładzie powyżej nakazujemy *scale* zmienić szerokość strumienia do 150 pikseli, dostosowując równocześnie wysokość z zachowaniem proporcji obrazu.

O którym jednak strumieniu mówimy? Zamieszczone wyżej polecenie ma dwa wejścia: *film.mp4* i *LM_logo.mp4*. Jak poinformować FFmpega, które wejście chcemy przeskalować? Do tego celu służy wyrażenie *[1:v]*. W napisie *filter_complex* możemy wybrać strumień, na którym będziemy przeprowadzać operacje za pomocą liczby i rodzaju strumienia. Wejścia rozpoczynają się od *0*, zatem drugie (*LM_logo.png*) to *1*. Litera informuje filtr, na którym strumieniu ma operować. PNG zawiera jedynie element wizualny (obraz), więc nakazujemy *scale*, by został użyty *[1:v]*. Zdarza się, że innego rodzaju wejścia mają więcej komponentów. Przykładowo wejście *example.mp4* składa się z komponentu wizualnego i dźwiękowego. Chcąc więc nałożyć filtry na dźwięk, użylibyśmy konstrukcji *[0:a]*; z kolei ścieżkę z napisami zmodyfikowalibyśmy za pomocą [0:s] itd.

Warto przy okazji wspomnieć, że FFmpeg umożliwia nadawanie etykiet strumieniom. Do tego właśnie służy *[ol]*: nakładamy filtr skalujący na *[1:v]*, a następnie nadajemy mu nazwę *[ol]*, dzięki czemu później będziemy mogli z niej korzystać. Nazwa może być dowolna.

Jak widzimy, w wierszu 4 Listingu 3 możemy użyć strumienia wideo z pierwszego wejścia (*[0:v]*) i nałożyć na niego przeskalowany obraz (*[ol]*). Przecinek oddzielający filtry *scale* i *overlay* oznacza, że dane wyjściowe z pierwszego zostaną przesłane potokiem do drugiego, czyli że zamiast:

```
[0:v] [ol] overlay=W-w-10:H-h-10
```

mogliśmy użyć konstrukcji:

```
[0:v] overlay=W-w-10:H-h-10
```

uzyskując identyczny rezultat.

Za pomocą *filter_complex* możemy tworzyć bardzo złożone wyrażenia. Odkryjemy wtedy, że etykiety są nie tyle przydatnym dodatkiem, ile nieodzownym elementem umożliwiającym zyskanie żądanego rezultatu.

Rysunek 2: Film oznaczony gigantycznym znakiem wodnym.

Rysunek 3: Możemy umieścić logo w dowolnym miejscu, przekazując *overlay* pozycję w postaci parametrów *x* i *y*.

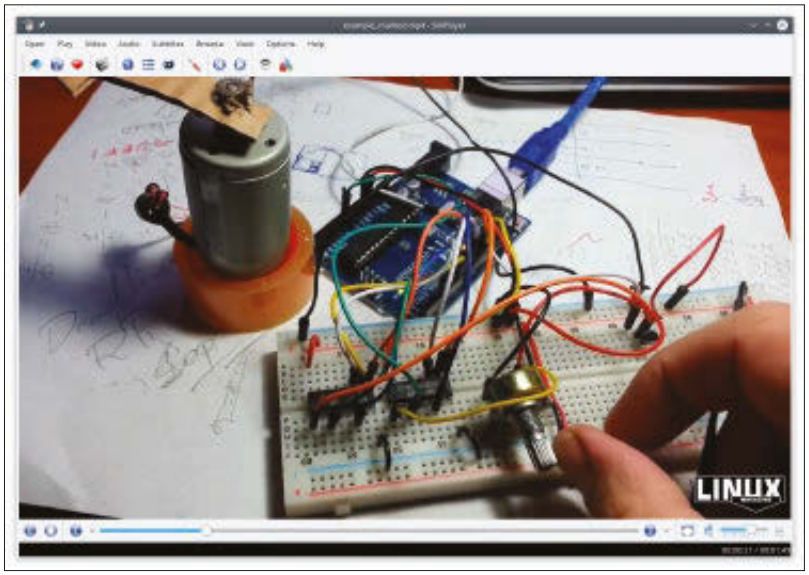
Ostatecznie strumień *LM_logo.png* zostanie zmniejszony do rozsądnych rozmiarów, a następnie nałożony na prawy dolny róg obrazu, jak to widzimy na Rysunku 4.

Subtelna mieszanka

Nakładka jest dobrym rozwiązaniem, jeżeli nie przeszkadza nam fakt, że nieprzezroczyste logo przesłania część obrazu. Jeśli jednak potrzebujemy rozwiązania, które umożliwi użytkownikom oglądanie całości, powinniśmy użyć półprzezroczystego logo.

W tym celu zamiast *overlay* użyjemy filtra *blend*. Umożliwia on łączenie warstw za pomocą różnych algorytmów. Możemy używać dodawania, odejmowania, XOR itd., przez co *blend* jest znacznie bardziej elastyczny niż *overlay*.

Jest jednak jeden haczyk: *blend* wymaga, by oba łączone strumienie miały te same proporcje (storage aspect ratio, SAR). We współczesnych formatach cyfrowych piksele są zazwyczaj idealnie



Rysunek 4: Logo zostało przeskalowane i umieszczone w odpowiednim miejscu za pomocą filtrów *overlay* i *scale*.

kwadratowe, więc SAR wynosi 1:1:. Jeśli nie jesteśmy pewni proporcji w danym klipie, możemy użyć polecenia *ffprobe* z pakietu FFmpeg:

```
ffprobe film.mp4
```

Zgodnie z oczekiwaniami powinniśmy uzyskać rozdzielczość 1280 × 720 pikseli i SAR 1:1.

Natomiast poniższe polecenie:

```
ffprobe LM_logo.png
```

wykaże rozdzielczość 800 × 314 pikseli i SAR 2515:2515.

Oznacza to, że musimy zwiększyć rozdzielczość górnej warstwy do 1280 × 720 pikseli. Choć 2515:2515 to to samo co 1:1, FFmpeg o tym nie wie, więc będziemy musieli sobie z tym poradzić.

Rysunek 5: Półprzezroczyste logo nałożone na strumień wideo.

LISTING 1: Nałożenie logo na film

```
ffmpeg -i film.mp4 -i LM_logo.png\
-filter_complex "\
overlay\
"\
 -codec:a copy film_oznaczony.mp4
```

LISTING 2: Nakładka i wybór miejsca

```
ffmpeg -i film.mp4 -i LM_logo.png\
-filter_complex "\
overlay=W-w-10:H-h-10\
"\
-codec:a copy film_oznaczony.mp4
```

LISTING 3: Skalowanie i nakładka

```
ffmpeg -i film.mp4 -i LM_logo.png\
-filter_complex "\
[1:v] scale=150:-1 [ol], \
[0:v] [ol] overlay=W-w-10:H-h-10\
" \
-codec:a copy film_oznaczony.mp4
```

Jeżeli przeskalujemy jedynie logo i zmienimy SAR za pomocą *setsar=sar=1* jak na Listingu 4, polecenie to zadziała, jednak wynik nie będzie zgodny z naszymi oczekiwaniami (Rysunek 5).

Choć to, co widzimy na Rysunku 5, nie odpowiada naszym oczekiwaniom, przeanalizujmy Listing 5, zawiera on bowiem kilka nowych i interesujących funkcji. Po pierwsze, zwróćmy uwagę na trzy bloki z łańcuchami filtrów w wierszach 3, 4 i 5, oddzielone od siebie dwukropkami. Dwukropek między blokami filtrów oznacza, że dany blok nie jest związany z następnym. Dlaczego? Ponieważ w wierszu 3 nakładamy filtry na logo (wejście 2), w wierszu 4 nakładamy filtry na film (wejście 1), w wierszu 5 zaś łączone są oba strumienie z nałożonymi filtrami.

W wierszu 3 logo (wejście 2) zostaje przeskalowane do 1280 × 720 pikseli, natomiast SAR zmienione do 1. Również w wierszu 4 dbamy o właściwe ustawienie SAR (*1*) i konwertujemy każda ramkę do przestrzeni kolorów RGBA, by można było je połączyć z logo bez żadnych dziwnych efektów kolorystycznych. Wreszcie w wierszu 5 używamy filtra *blend* do przetworzenia wyjścia z wierszy 3 i 4 oraz konwersji połączonych warstw na przestrzeń kolorów używaną w filmach.

Filtr *blend* może przyjąć więcej niż jeden parametr – a nawet kilkanaście [3]. Zamiast więc umieszczać parametry po kolei jeden za drugim, oddzielając je dwukropkiem jak w przypadku *scale*, powinniśmy raczej jawnie używać nazwy parametru, by uniknąć niejasności. Robimy to, łącząc nazwę każdego parametru z nazwą jak na Rysunku 6.

LISTING 4: Półprzezroczyste logo

```
ffmpeg -i film.mp4 -i LM_logo.png \
-filter_complex "\
[1:v] scale=1280:720, setsar=sar=1 [lo];\
[0:v] setsar=sar=1, format=rgba [bg];\
[bg][lo] blend=all_mode=addition:all_opacity=0.5,
format=yuva422p10le\
"\
-codec:a copy film_oznaczony.mp4
```

LISTING 5: Zmiana pozycji i rozmiaru

```
ffmpeg -i film.mp4 -i LM_logo.png\
-filter_complex "\
[1:v] scale=150:-1, pad=1280:720:ow-iw-10:oh-ih-10,
setsar=sar=1, format=rgba [lo];\
[0:v] setsar=sar=1, format=rgba [bg];\
[bg][lo] blend=all_mode=addition:all_opacity=0.5,
format=yuva422p10le\
"\
-codec:a copy film_oznaczony.mp4
```

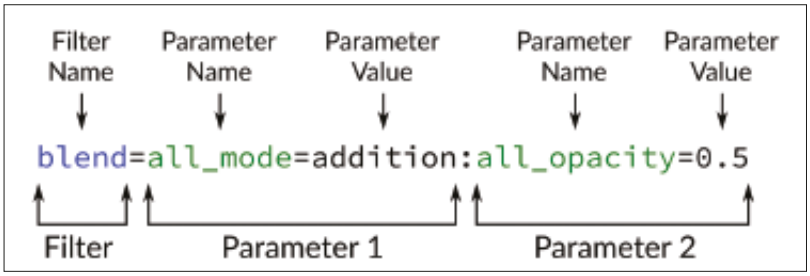
W tym przypadku przekazujemy *blend* opcję, która informuje filtr, w jaki sposób należy połączyć oba piksele, używając w tym celu parametru *all_mode* i jaka ma być przezroczystość każdego piksela (0.5 oznacza 50% przezroczystości), używając w tym celu parametru *all_opacity*.

W ten sposób nauczyliśmy się korzystać z etykiet, opisując nimi filtrowane wejście: *[bg]* to tło, natomiast *[lo]* to nałożone logo.

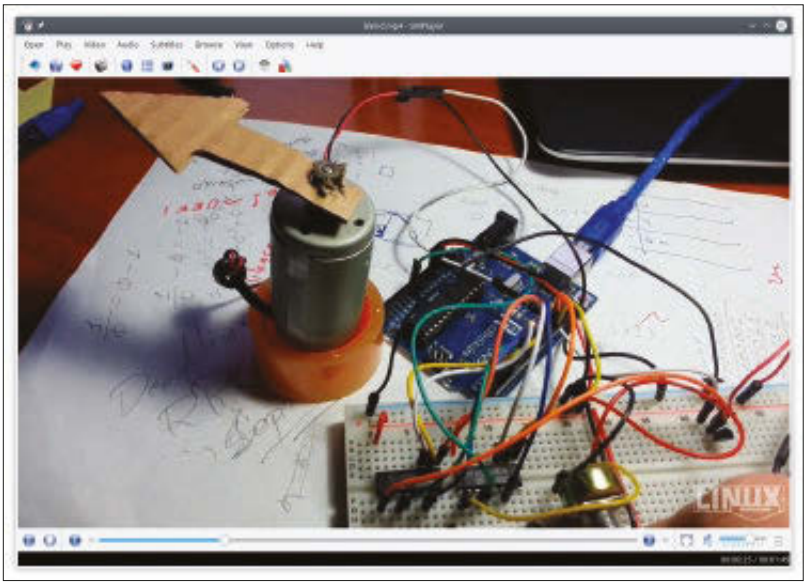
Jak jednak wspomnieliśmy, nie jest to jeszcze żądany rezultat. Chcemy, aby logo znalazło się w prawym dolnym rogu, a nie przesłaniało cały ekran. Na szczęście FFmpeg zawiera kolejny filtr, który pomoże nam osiągnąć cel: *pad* [4] umożliwia zmianę wymiarów ramki, wypełniając pustą przestrzeń żądanym kolorem bądź przezroczystością alfa (Listing).

Wszystkie zmiany odbywają się w wierszu 3, w którym zmniejszamy logo, używając w tym celu znanego nam już filtra *scale*. Następnie tworzymy przestrzeń dookoła, by ramka miała 1280 × 720 pikseli. Podobnie jak było to w przypadku *overlay*, możemy zdecydować, w którym miejscu ramki umieścić oryginalny obraz – albo za pomocą liczb, albo używając wbudowanych zmiennych: *ow* (szerokość ramki), *iw* (szerokość oryginalnego obrazu), *oh* (wysokość ramki) oraz *ih* (wysokość oryginalnego obrazu). Podobnie jak jest to w przykładzie z *overlay*, *ow-iw-10:oh-ih-10* umieszcza logo 10 pikseli na lewo od prawej krawędzi ramki i 10 pikseli nad dolną krawędzią.

Na koniec konwertujemy przestrzeń kolorów ramki do *rgba* filtrem *format*, by nic dziwnego nie przydarzyło się kolorom podczas łączenia obu warstw. Wynik działania polecenia widzimy na Rysunku 7 – tym razem powinien on odpowiadać naszym oczekiwaniom.



Rysunek 6: Przekazywanie filtrowi kilku parametrów wraz z ich nazwami.



Rysunek 7: Nałożone na obraz logo wygląda teraz bardziej profesjonalnie.

Wnioski

Zaprezentowane w niniejszym artykule mogą wyglądać na skomplikowane, FFmpeg zaś może odstraszać złożoną składnią. Jest tak po części dlatego, że edycja filmów sama w sobie nie jest prostą sztuką. Warto jednak podjąć wyzwanie, korzyści bowiem są ogromne. Przekazując FFmpegowi wykonywanie prostych, powtarzalnych czynności, możemy uniknąć konieczności instalacji dużych aplikacji i marnowania czasu na ręczne wykonywanie prac, którymi z powodzeniem może zająć się automat.

FFmpeg jest obecnie dojrzałym oprogramowaniem i pracuje się na nim znacznie bardziej stabilnie niż na większości edytorów wideo, unikniemy więc frustracji związanej z zawieszającymi się programami. Ponieważ jest to narzędzie działające w wierszu poleceń, umożliwia przetwarzanie wielu filmów po kolei bez konieczności dodatkowej interwencji z naszej strony czy nawet nadzorowania całego procesu. Nie bez znaczenia jest również fakt, że FFmpeg bez problemu zadziała na nieposiadającym monitora serwerze, dzięki czemu naszej stacji roboczej możemy użyć do bardziej przydatnych zadań, takich jak granie w gry czy przeglądanie witryn internetowych.

Jakkolwiek by nie patrzeć, edycja filmów za pomocą FFmpega przynosi same korzyści.

INFO

[1] Witryna FFmpega: http://ffmpeg.org/
[2] Lista filtrów FFmpega: https://ffmpeg.org/ffmpeg-filters.html
[3] Parametry filtra *blend*: https://ffmpeg.org/ffmpeg-filters.html#blend_002c-tblend
[4] Filtr *pad*: https://ffmpeg.org/ffmpeg-filters.html#pad-1

Numer 169/Marzec 2018

Kompilatory

Trudno byłoby tworzyć oprogramowanie bez niezwykłego narzędzia, które pośredniczy między ludźmi a komputerami. W przyszłym miesiącu przyjrzymy się bliżej kompilatorom.

Ilustracja © pratyaksa, 123RF.com