i weryfikujemy jego poprawność – najłatwiej, dodając je po prostu do systemu. Dla uproszczenia repozytorium dodajemy na tym samym komputerze, na którym tworzymy je w przykładzie:

```
sudo zypper addrepo -f ➥
http://localhost/repo local
```

Opcja *-if* odpowiada za automatyczne odświeżanie repozytorium przez system. Jeśli jej nie dodamy, będziemy musieli za każdym razem wykonywać to ręcznie:

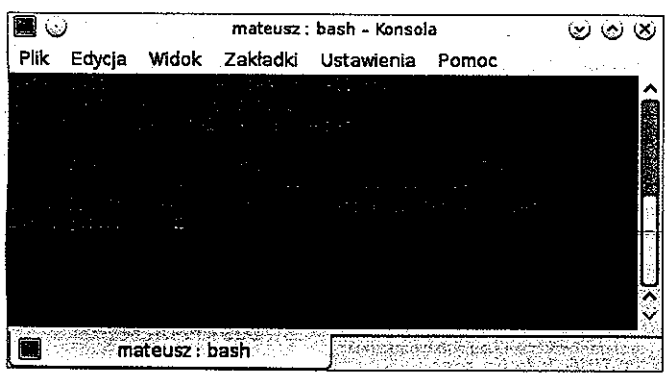
```
sudo zypper refresh
```

Jeżeli repozytorium jest dodane do listy, wszystko działa, jak należy.

Konfiguracja Jenkinsa

Ostatni etap to połączenie Jenkinsa, systemu *subversion* i naszego repozytorium paczek RPM.

Dodajemy nowy projekt o nazwie *helloRPM*; jako repozytorium kodu źródłowego wskazujemy *subversion*. W polu URL repozytorium SVN piszemy *svn://localhost/*. W sekcji *build* dodajemy nowy etap budowy projektu.



Rysunek 2: Wyszukiwanie paczki w repozytoriach

W tym celu klikamy *Add build step -> Execute shell*, jako wartość podając zawartość Listingu 3.

Skonfigurowany projekt zapisujemy i próbujemy uruchomić zadanie zbudowania paczki. Jeśli kończy się ono poprawnie, możemy wyszukać paczkę (Rysunek 2), po czym zainstalować ją poleceniem:

```
sudo zypper in HelloRPM
```

Jeśli paczka jest zainstalowana, powinna bez problemów dać się uruchomić:

```
/usr/bin/hello.sh
```

Od tej pory za każdym razem, gdy Jenkins zbuduje nową paczkę o wyższym numerze wersji (pola *version* i *release* pliku z Listingu 2), jej aktualizacja na komputerach będzie możliwa poprzez wykonanie polecenia:

```
sudo zypper update HelloRPM
```

Problemy

Gdyby coś poszło nie tak, powinniśmy się upewnić, że wszystko mamy wdrożone jako usługi udostępnione na komputerze lokalnym. Jeśli wdrożenie wykonaliśmy na kilku maszynach, problemem mogą być ustawienia zapór sieciowych. Upewniamy się również, że wszystkie niezbędne usługi, takie jak serwer HTTP, *subversion* czy Jenkins, są uruchomione / włączone.

Podsumowanie

Takie wdrożenie systemu ciągłej integracji zapewnia zawsze aktualne oprogramowanie na systemach użytkowników.

W zależności od potrzeb zamiast repozytorium oprogramowania możemy skonfigurować repozytorium aktualizacji systemu, aby zapewnić ciągłość aktualizacji bezpieczeństwa i nie tylko. ■■■

LISTING 3: Konfiguracja Jenkinsa

```
dirs=("BUILD" "RPMS" "SOURCES" ➥
  "SPECS" "SRPMS")
for dir in "${dirs[@]}"
do
  if [[ -d $dir ]]
  then
    rm -Rf $dir
  fi
  mkdir $dir
done
mv *.sh SOURCES/.
mv *.spec SPECS/.
path=`pwd`
rpmbuild --define ➥
"_topdir $path" ➥
-ba SPECS/hello.spec
cp ./RPMS/noarch/*.rpm ➥
/srv/www/htdocs/repo/noarch/
createrepo ➥
/srv/www/htdocs/repo/
```

AUTOR

Mateusz Lach, z wykształcenia informatyk, od ponad sześciu lat zajmuje się wytwarzaniem oprogramowania. Jest autorem kilku publikacji w „Linux Magazine" oraz książki *Bash. Praktyczne skrypty*. Od trzech lat pracuje jako programista w międzynarodowym koncernie energetycznym Vattenfall.

Wykrywanie intruzów przy użyciu Suricaty

Na celowniku

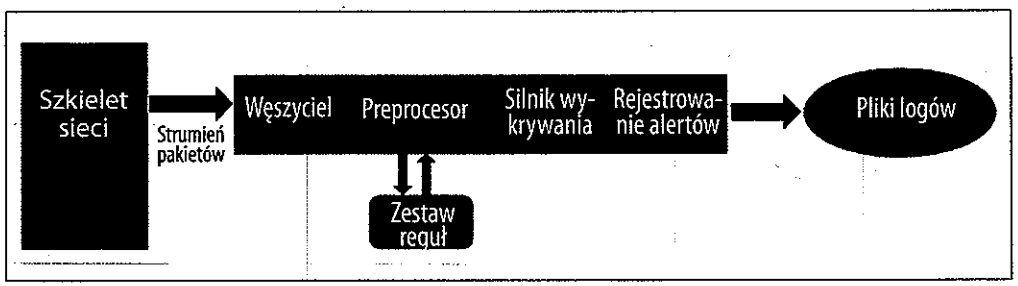
Kiedyś napastnicy sieciowi przypominali kieszonkowców czy ulicznych bandytów: znajdowali to, co chcieli, zabierali i uciekali. Dzisiejsze zaawansowane trwałe zagrożenia (ang. ***Advanced Persistent Threats***, APT) są znacznie bardziej przebiegłe i niebezpieczne. Intruz może znaleźć sposób, aby wykraść naszą tożsamość i długo pozostać w systemie (patrz: ramka „Czas w systemie"). Chcąc się go pozbyć, powinniśmy zastosować inne, bardziej metodyczne podejście. *James Stanger*

Dla wielu administratorów system wykrywania i zapobiegania włamaniom Snort (IDS/IPS) [1] pozostaje pierwszą linią obrony. Istnieją jednak także alternatywne rozwiązania – niektóre oferują różne korzyści w pewnych sytuacjach. Artykuł opisuje narzędzie Suricata [2], otwartoźródłowy system IDS/IPS, za którym stoi Open Information Security Foundation (OSIF) [3]. Może ono odczytywać zestawy reguł Snorta, co ułatwia obsługę obu systemów, a nawet migrację – jeśli uznamy, że do naszego środowiska lepiej jest przystosowane jedno z nich.

Dlaczego Suricata?

Na stronie głównej Suricaty programiści wymieniają trzy powody, dla których warto wypróbować to narzędzie:

- wysoka skalowalność – narzędzie jest wielowątkowe, co maksymalizuje wydajność systemu skonfigurowanego jako jego czujnik; według programistów Suricata może przesyłać dane z prędkością 10 Gbit,
- rozpoznawanie protokołów – Suricata automatycznie rozpoznaje najpopular-



© Kurhan, 123RF.com

Rysunek 1: System przetwarzania pakietów Snorta.

Rysunek 2: System przetwarzania pakietów Suricaty.

niejsze protokoły natychmiast po rozpoczęciu strumienia, umożliwiając pisanie reguł dla protokołu, a nie dla oczekiwanego portu,
- rozpoznawanie plików, sumy kontrolne MD5 i ekstrakcja plików – narzędzie potrafi rozpoznawać tysiące rodzajów plików; oblicza sumy kontrolne MD5 w locie, umożliwiając monitorowanie zmian w plikach systemowych lub wykrywanie złośliwych plików ze znanymi wartościami sum kontrolnych.

Na początek najlepsze jest szybkie porównanie z popularnym i równie potężnym narzędziem Snort. Suricata ma całkowicie inną bazę kodu, mimo że obsługuje reguły Snorta. Na Rysunku 1 widzimy, jak postępuje podczas wykrywania ruchu. Silniki węszyciela, preprocesora i wykrywania są całkowicie odrębne.

Zwróćmy szczególną uwagę, jak Snort rozdziela funkcje węszyciela i preprocesora, które pozwalają mu szybko zebrać dane na temat sieci, a następnie je przetworzyć, zanim zastosują reguły. Funkcja zestawu reguł jest osobna.

Jest dobry powód, aby rozdzielić te funkcje: funkcja preprocesora umożliwia Snortowi szybkie identyfikowanie określonych strumieni ruchu. W przeciwnym razie stosowanie reguł do niektórych strumieni ruchu zajęłoby sporo czasu i szybko doprowadziło przytłoczenia Snorta nawet przy umiarkowanych ilościach ruchu sieciowego.

Na Rysunku 2 widzimy schemat systemu przetwarzania pakietów Suricaty. Zwróćmy uwagę na różnice: podczas gdy Snort rozdziela funkcje węszyciela i preprocesora, Suricata łączy proces podczas pozyskiwania pakietów. Jak widzimy, Suricata zawiera osobną funkcję dekodowania umożliwiającą kontrolowanie strumienia danych na poziomie warstwy aplikacji.

Następnie narzędzie może wykorzystać łańcuch ciągów wykrywania. Wynik jest wysyłany do systemu monitorującego sieć lub do bazy danych. Suricata stosuje bardziej warstwowe podejście do odczytywania i przesiewania informacji.

Zarówno Snort, jak i Suricata mają swoje zalety, ale w niektórych sytuacjach Suricata może się okazać lepszym rozwiązaniem.

Instalacja Suricaty

Podobnie jak Snort, Suricata to otwartoźródłowy system wykrywania intruzów, który potrafi też zapobiegać włamaniom. Innymi słowy, nie tylko ujawnia ataki, ale może także zakłócać ruch, który uznaje za podejrzany. Ma stabilny kod i pozostaje w powszechnym użyciu. Nie jest wprawdzie tak popularna, jak Snort, ale może monitorować ruch zarówno IPv4, jak i IPv6, a także wykrywać problemy związane z bezpieczeństwem. Dość elastyczna, działa na wszystkich wariantach Linuksa. Aktualna wersja Suricaty ma numer 2.07.

Aby zainstalować Suricatę z funkcjami wykrywania i zapobiegania włamaniom, instalujemy biblioteki wspierające Libp:

```
$ sudo apt-get -y install libpcre3
libpcre3-dbg libpcre3-dev \
build-essential autoconf
automake libtool libpcap-dev
libnet1-dev \
libyaml-0-2 libyaml-dev
zlib1g zlib1g-dev libcap-ng-dev
libcap-ng0 \ make libmagic-dev
```

po czym możemy pobrać Suricatę z Sieci:

```
$ wget
http://www.openinfosecfoundation.org/
download/suricata-2.0.7.tar.gz
tar -xvzf suricata-2.0.7.tar.gz
cd suricata-2.0.7
```

Następnie kompilujemy pliki źródłowe do pliku binarnego:

```
./configure --prefix=/usr
--sysconfdir=/etc --localstatedir=/var
make
sudo make install
sudo ldconfig
```

W niektórych sytuacjach konieczne może być ręczne utworzenie katalogu dla logów:

```
$ sudo mkdir /var/log/suricata
$ sudo mkdir /etc/suricata
```

Upewniamy się, że katalogi należą do procesu Suricaty, a także do konta *root:*, i kopiujemy pliki do odpowiednich katalogów:

```
$ sudo cp classification.config
/etc/suricata
$ sudo cp reference.config /etc/suricata
$ sudo cp suricata.yaml /etc/suricata
```

W ten sposób mamy działającą Suricatę, na razie jednak bez zainstalowanych reguł. Jeśli chcemy, możemy je zaczerpnąć ze Snorta, uzyskać od współpracowników lub ze źródeł zewnętrznych. Możemy też pobrać domyślne pliki reguł projektu *Emerging Threats* – wprawdzie nie są szczególne, ale mogą posłużyć za dobrą podstawę do tworzenia własnych. Chcąc pobrać reguły domyślne, piszemy:

```
$ wget U
http://rules.emergingthreats.net/
open/suricata/emerging.rules.tar.gz
tar zxvf emerging.rules.tar.gz
cp -r rules /etc/suricata/
```

Rysunek 3: Weryfikowanie instalacji Suricaty.

Następnie, po skopiowaniu reguł, uruchamiamy Suricatę, aby sprawdzić, czy działa. Jeśli na przykład mamy typowy interfejs *eth0*, uruchamiamy ją poleceniem:

```
$ sudo suricata -c
/etc/suricata/suricata.yaml -i eth0
```

po czym za pomocą *suricatasc* upewniamy się, że działa (Rysunek 3).

Wynik polecenia z Rysunku 3, użytego z opcją *c*, pokazuje informacje o trybie, w którym uruchomiona jest Suricata. Szczególnie ważna jest opcja *running mode*, sygnalizująca, że program działa poprawnie; *capture mode* informuje, czy Suricata przechwytuje pakiety. Wartość *AFPACKET_DEV* pokazuje, czy Suricata działa, a *ifacestat* informuje o statusie interfejsów Suricaty:

```
$ sudo suricatasc -iface-stat eth0
"drop" 2136
„invalid checksum": 0,
„pkts: 80595950"
```

Z danych wyjściowych wynika, że porzuconych zostało 2136 pakietów z 80 595 950 i że żadne pakiety nie mają nieprawidłowych sum kontrolnych.

Rozwiązywanie problemów z implementacją

Jednym z najczęstszych problemów podczas instalowania Suricaty są nierozwiązane zależności dotyczące *libpcre3*. Aby tego uniknąć, najlepiej wybrać Linuksa z zainstalowaną już Suricatą lub przynajmniej przygotowanego do zaimplementowania narzędzia. Najlepiej sprawdzają się dystrybucje bazujące na Debianie, takie jak Ubuntu. Powinniśmy się też upewnić się, że uruchomione są wszystkie wymagane usługi:

```
$ sudo service barnyard2 start
$ sudo service snortd start
$ sudo service httpd start
```

Tworzenie własnego systemu

Aby dostosować Suricatę do naszych potrzeb, edytujemy plik */etc/suricata/suricata.yaml* za pomocą dość prostej składni:

```
alert icmp any any -> 
any any (msg:"wykryto ping";
sid:2; rev:1;)
```

która definiuje alert dla ruchu ICMP, nakazując Suricacie wyszukiwać dowolny ruch ICMP z dowolnego interfejsu. Po wykryciu takiego ruchu, do plików dziennika trafia alert o treści *PING detected.*

Podczas analizowania ruchu sieciowego ważne, aby najpierw zrozumieć charakter firmy, skupiając się na celu działalności. W przypadku firmy detalicznej serwery i bazy danych e-commerce powinny być odpowiednio chronione. Jeśli zaś firma ma serwer przetwarzający dane z kart kredytowych, stworzenie reguł Suricaty do wykrywania niezaszyfrowanych transakcji przy użyciu kart kredytowych pozwala się upewnić, że informacje są szyfrowane we właściwy sposób. Ważne również, aby móc się dowiedzieć, czy intruz zdołał uzyskać dostęp do serwera e-commerce i wysyła niezaszyfrowane dane do serwera spoza sieci. Aby stworzyć taką regułę, możemy – otworzywszy dowolny edytor tekstowy – umieścić w pliku */etc/suricata/suricata/yaml* kod:

```
alert tcp any any <>
any any (pcre:"/5\d{3}(\s|-)?
\d{4}(\s|-)?\d{4}(\s|-)?\d{4}/"; \
    msg:"MasterCard number U
        detected in clear text";content:U
        "mastercard";nocase;sid:
        9000001;rev:1;)
```

Przykładowa reguła sprawdza dane z kart MasterCard; może dotyczyć również kart American Express:

```
alert tcp any any <>
any any (pcre:"/3\d{3}U
(\s|-)?\d{6}(\s|-)?\d{5}/";
msg:"American Express card number
detected in clear text";content:
„amex";nocase;sid: 9000003;rev:1;)
```

CZAS W SYSTEMIE

W większości firm próg poziomu bezpieczeństwa wynosi 80%. Osiągnięcie poziomu 90% wymaga ciężkiej pracy i mądrej alokacji zasobów. Częściowym rozwiązaniem jest stworzenie własnego systemu wykrywania intruzów. Dodatkowo powinniśmy też znać cykl działań intruza:

1. Atak wstępny: intruz jest w stanie uzyskać dostęp do systemu.
2. Dalsze ruchy: intruz przenosi się z jednego systemu do drugiego, zwykle poprzez wykorzystywanie niezabezpieczonych systemów lub relacje zaufania między systemami.
3. Eksfiltracja: intruz przenosi dane z sieci wewnętrznej do zdalnej.

Kluczem do przerwania go jest ograniczenie czasu pozostawania intruza w systemie, czyli czasu między atakiem wstępnym a wykryciem. Dość powszechnie za najważniejsze uchodzi skupienie się na liczbie znalezionych i usuniętych podatności. Jednemu napastnikowi wystarczy jednak tylko jedna luka, aby dostać się do systemu; czas jego pozostawania w systemie często znacznie lepiej sygnalizuje poziom krytyczności ataku i zagrożenia w sieci. Klucz do ograniczenia czasu pozostawania intruza w systemie i stworzenia odpornej strategii obronnej jest prosty: ochrona danych w miejscu ich pozostawania. Tworząc własny system, powinniśmy się skupić na ścieżkach do tych informacji.

Jeśli najważniejsze jest skrócenie czasu pozostawania intruza w systemie i wykrycie go, powinniśmy się skupić na:

- tunelowanym ruchu, który pochodzi z zaufanego systemu, ale trafia do nieznanej sieci; często jest szyfrowany,
- podejrzanym ruchu przesyłanym przez długi czas; napastnik, chcąc zaatakować system, wcześniej może dążyć do poznania wszystkich zasobów - nawet przez kilka tygodni, aby nie zostać wykrytym - przy niewielkim ruchu tak, aby jego działania nie przekraczały zdefiniowanych progów detekcji.

Reguła:

```
alert udp $EXTERNAL_NET any
-
> $HOME_NET 53 (msg:"Potential NSTX DNS
Tunneling";
content:"|01 00|"; offset:2; w
ithin:4; content:"cT";
offset:12; depth:3; content:"|00 10 00 01|"; within:255; class
type:bad
-
unknown; sid:1000 2;
```

może posłużyć do wykrycia ruchu tunelowanego w ruchu zapytania DNS przy wykorzystaniu portu UDP 53. Analizuje ruch DNS pod kątem danych dotyczących serwera e-commerce lub bazy danych.

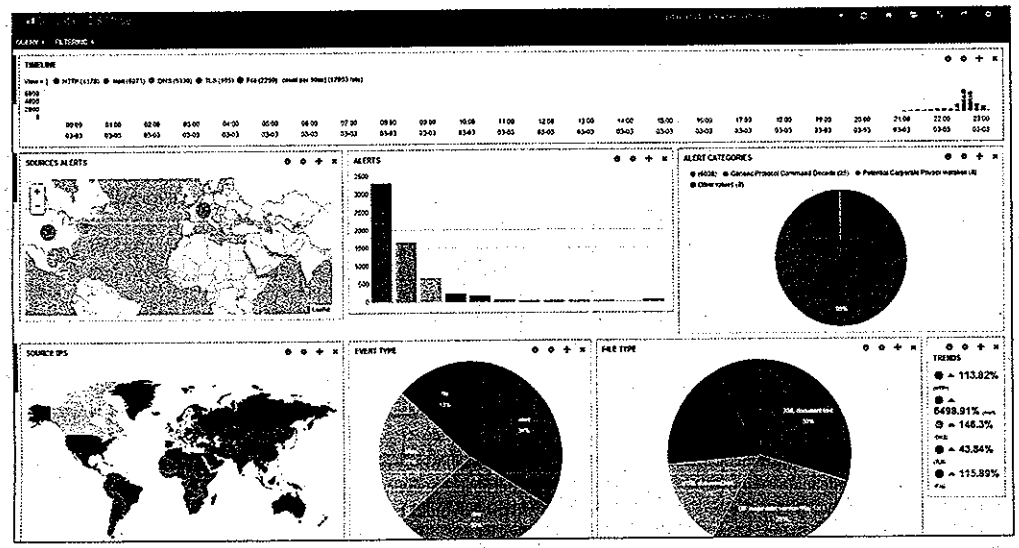
Jak może zauważamy, nie są to zestawy reguł Snorta. Powyższe rozpoczynają tworzenie systemu wykrywania nieautoryzowanego dostępu w oparciu o elastyczny system dostosowany do naszych potrzeb; nie musimy na tym poprzestawać.

Używanie Suricaty

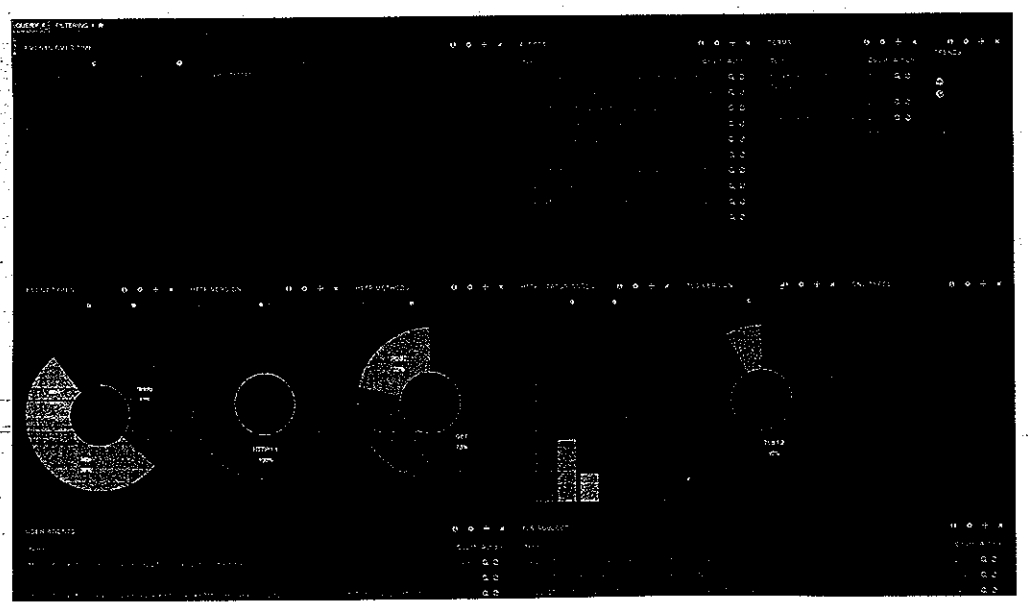
Gdy Suricata zaczyna działać, możemy się zalogować i wyświetlać alerty dotyczące ruchu.

Na Rysunku 4 znajdują się wyniki działań Suricaty związanych z wykrywaniem problemów z ruchem sieciowym i DNS. Raport pozwala sprawdzić rodzaj i źródła ruchu. Mimo że hakerzy mogą fałszować adresy IP, nadal warto spróbować poznać miejsce ruchu.

Przedstawiony na Rysunku 5 panel pokazuje zdarzenia związane z bezpieczeństwem w czasie; przeglądanie ich pozwala wykryć, kiedy intruz mapował sieć lub określony zasób sieciowy.

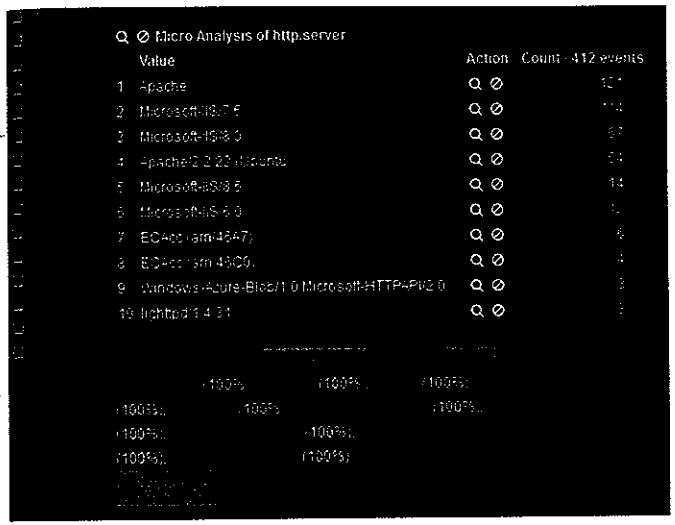


Rysunek 4: Suricata w tradycyjnym trybie IDS.



Rysunek 5: Zdarzenia w czasie.

Wreszcie Rysunek 6 to raport na temat rodzajów usług internetowych dostępnych w sieci, przydatny do wykrywania starszych serwerów, które często padają ofiarami ataków. Do filtrowania informacji hakerzy często wykorzystują dostępne gotowe programy, jak IIS czy Apache Server. Możliwość wykrywania serwerów ułatwia rozpoznawanie takiego ruchu.



Rysunek 6: Wykrywanie serwerów przy użyciu Suricaty.

Podsumowanie

Suricata to praktyczna alternatywa Snorta i innych systemów wykrywania i zapobiegania włamaniom. Jeśli poszukujemy wysoce skalowalnego systemu wykrywania intruzów, który jest łatwy do skonfigurowania i automatycznie rozpoznaje wiele protokołów i typów plików, z pewnością warto rozważyć użycie Suricaty. ■■■

INFO

[1] Snort: https://www.snort.org/
[2] Suricata: http://suricataids.org
[3] Open Information Security Foundation: http://oisf.net