Obrona

ClamAV, czyli jak to robią małże...

Tomasz Kojm

stopień trudności

internetowe nastawione na kradzież danych bądź transformację komputera ofiary w maszynkę do robienia pieniędzy, to już dzisiaj standard. Wieloletnia wojna z nimi pokazała, że najlepszą strategią jest bezpośrednia eliminacja zagrożeń na poziomie serwerów pocztowych. Tu z pomocą przychodzi oprogramowanie Open Source, które oferuje gamę rozwiązań do walki ze spamem i przynajmniej jeden program do bezpośredniej walki ze złośliwym oprogramowaniem, który zostanie opisany w tym artykule.

Problem zalewu niechcianej poczty dotyczy nas wszystkich, zarówno administratorów systemów, jak i zwykłych użytkowników Internetu, który w ostatnich latach stał się łakomym kąskiem dla szeroko pojętej branży marketingowej oraz wszelkiej maści przestępców, zainteresowanych głównie kilkoma numerami z naszych kart kredytowych. Ataki typu *phishing* czy robaki

Projekt Clam AntiVirus (ClamAV) to zestaw narzędzi antywirusowych przeznaczonych dla systemów uniksowych. Rozwijany jest przez międzynarodową grupę kilkunastu deweloperów i dostępny na licencji GNU GPL v2. Należy w tym miejscu zwrócić uwagę na to, co wyróżnia go na tle innych programów Open Source, czyli na potrzebę ciągłej aktualizacji – zarówno kodu, jak i baz sygnatur wirusów. W przypadku programów antywirusowych, to właśnie te elementy: oprogramowanie i usługa, odgrywają jednocześnie kluczową rolę.

Słowo *clam* w języku angielskim oznacza małża. Czytelnicy, którym zdarzało się bywać na lekcjach biologii, być może zapamiętali, że małże należą do zwierząt odżywiających się metodą filtrowania: aby przeżyć, muszą w efektywny sposób filtrować wodę w poszukiwaniu cząsteczek pożywienia. To krótkie wyjaśnienie powinno zaspokoić ciekawość dotyczącą nazwy programu.

Podstawowe narzędzia

Oprogramowanie ClamAV stosuje się do znanej uniksowej reguły KISS (ang. *Keep it Simple, Single*), stąd w wachlarzu programów dostarczanych w pakiecie źródłowym znajdują się proste w użyciu i wyspecjalizowane aplikacje przeznaczone dla konsoli, na bazie których można budować złożone systemy skanujące.

Clamscan

Poręczne narzędzie umożliwiające skanowanie plików oraz katalogów bezpośrednio z linii poleceń to *clamscan*. W najprostszym przypadku, jego użycie sprowadza się do wskazania nazwy

Z artykułu dowiesz się...

- jak zbudowany jest Clam AntiVirus,
- w jaki sposób wykorzystać bibliotekę libclamav,
- w jaki sposób stworzyć własną sygnaturę wirusa dla ClamAV.

Co powinieneś wiedzieć...

- powinieneś posiadać podstawową wiedzę na temat zagrożeń płynących z Internetu,
- powinieneś posiadać wiedzę o systemach uniksowych oraz świadczonych przez nie usługach.

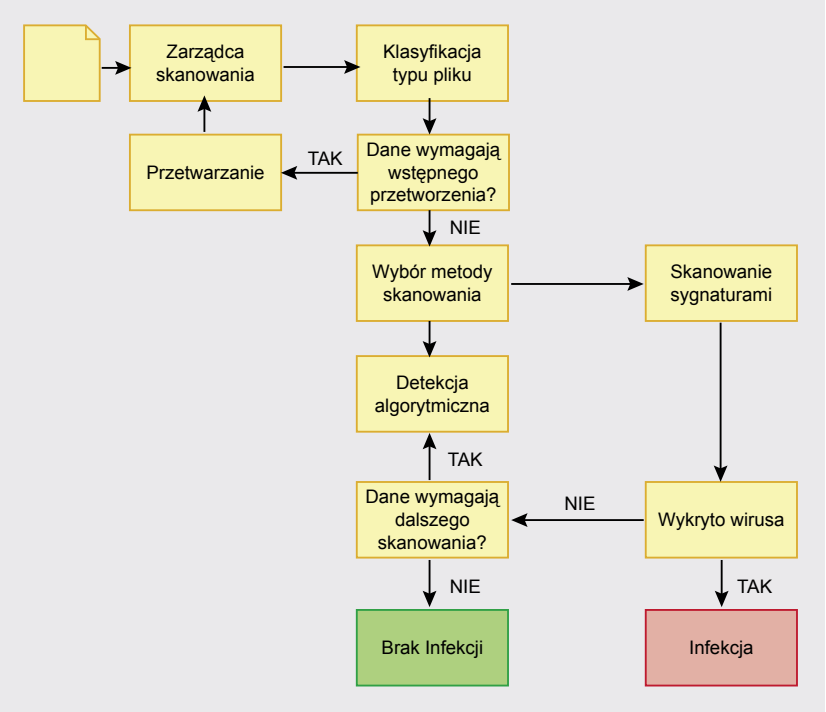
pliku bądź katalogu, który powinien zostać przeskanowany. Dzięki możliwości wywoływania zewnętrznych programów, potrafi sprawdzać archiwa, które nie są obsługiwane przez bibliotekę *libclamav*. *Clamscan* może zostać wykorzystany do wykonywania rutynowych zadań, na przykład skanowania katalogów użytkowników, bądź integracji z innymi programami, w celu wzbogacenia ich o możliwość wykrywania wirusów. Należy jednak podkreślić, że *clamscan* przy każdym uruchomieniu musi wczytywać pełne bazy sygnatur, co zajmuje zarówno dodatkowy czas, jak i pamięć operacyjną. Z tego powodu, nie powinien być stosowany w zadaniach, które mogą wymagać skanowania wielu plików z osobna i w tym samym czasie, takich jak skanowanie poczty na poziomie demona SMTP.

Clamd

Clamd to demon zaprojektowany do zadań, w których liczy się wydajność. Program ma budowę wielowątkową, dzięki czemu potrafi wykorzystać dodatkowe procesory w systemach SMP. Zapewnia ona także efektywne wykorzystanie baz sygnatur – są one ładowane przy starcie oraz po wykryciu nowej wersji, a następnie współdzielone przez wszystkie wątki. Po uruchomieniu, *clamd* przechodzi w tło, oczekując na polecenia (takie jak SCAN, PING, RELOAD, STREAM) przez wskazane w pliku konfiguracyjnym gniazdo uniksowe i/lub TCP. W przypadku systemów Linux oraz FreeBSD, potrafi dodatkowo współpracować z modułem Dazuko, który umożliwia skanowanie wybranych zasobów w momencie dostępu do nich. Pozwala to na przykład na monitorowanie zawartości publicznych katalogów FTP, gdy sam demon FTP nie posiada wsparcia dla skanowania antywirusowego.

Clamdscan

Jednym z narzędzi pozwalających na wykorzystanie dobrodziejstw demona *clamd* jest *clamdscan*. Nazwa sugeruje, że został on stworzony na podobieństwo pierwszego opisanego narzędzia i tak w rzeczywistości jest. *Clamdscan* może w wielu przypadkach efektywnie zastępować *clamscana*. Efektywnie, ponieważ nie potrzebuje on za każdym razem ładować baz sygnatur, co znacznie skraca czas uruchamiania, a także redukuje zużycie pamięci. *Clamdscan* nie zawiera żadnych procedur skanujących i opiera się wyłącznie na *clamd*, przesyłając do niego nazwy plików i katalogów, strumienie danych oraz deskryptory, które mają zostać przeskanowane. Z zależności tej wynika pewne istotne ograniczenie: w przypadku skanowania plików bądź kata-



Rysunek 1. *Uproszczony schemat działania biblioteki libclamav*

Listing 1. *Algorytmiczna detekcja wirusa wykorzystująca prostą kryptoanalizę*

```
/* W32.Parite.B */
if(SCAN_ALGO && !dll && ep == EC32(section_hdr[nsections -
                    1].PointerToRawData)) {
  lseek(desc, ep, SEEK_SET);
  if(cli_readn(desc, buff, 4096) == 4096) {
     const char *pt = cli_memstr(buff, 4040,
                                 "\x47\x65\x74\x50\x72\x6f\x63\x41\x64\x64\
                    x72\x65\x73\x73\x00", 15);
   if(pt) {
   uint32_t dw1, dw2;
   pt += 15;
   if(((dw1 = cli_readint32(pt)) ^ (dw2 = cli_readint32(pt + 4))) == 0x505a4f
                    &&
   ((dw1 = cli_readint32(pt + 8)) ^ (dw2 = cli_readint32(pt + 12))) ==
                    0xffffb &&
   ((dw1 = cli_readint32(pt + 16)) ^ (dw2 = cli_readint32(pt + 20))) == 0xb8)
                    {
   *ctx->virname = "W32.Parite.B";
   free(section_hdr);
   return CL_VIRUS;
    }
   }
  }
}
```

logów, clamdscan jest w stanie skanować tylko te, do których dostęp ma demon *clamd*.

Freshclam

W pakiecie ClamAV znajduje się program o wdzięcznej nazwie *freshclam*, który w pełni automatyzuje proces aktualizacji baz sygnatur wirusów. Program udostępnia szereg opcji: może być uruchamiany na żądanie bądź pracować w tle jako demon, potrafi współpracować z serwerami proxy, umożliwia wywołanie zewnętrznych programów w zależności od zdarzenia (aktualizacja baz, błąd, wykrycie nowej wersji oprogramowania). W celu sprawdzenia aktualnej wersji baz sygnatur, *freshclam* wykorzystuje DNS, a dokładniej specjalny rekord tekstowy:

```
$ host -t txt current.cvd.clamav.net
current.cvd.clamav.net TXT "0.88.4:
39:1640:1155043741:1"
```

Rekord ten zawiera także informację o aktualnej stabilnej wersji ClamAV, dzięki czemu *freshclam* może alarmować administratora o pojawieniu się nowego wydania programu.

Tabela 1. *Tabela znaków zastępczych*

Znak zastępczy	Znaczenie dopasowania
*	Dowolna ilość dowolnych znaków
??	Pojedynczy dowolny znak
{n}	n dowolnych znaków
{-n}	Co najwyżej n dowolnych znaków
{n-}	Co najmniej n dowolnych znaków
(a|b|c)	a lub b lub c

Tabela 2. *Tabela znaków zastępczych*

Kod pliku docelowego	Typ pliku
0	Dowolny
1	Portable Executable
2	Komponent OLE2 (na przykład skrypt VBA)
3	Znormalizowany HTML
4	Pocztowy
5	Graficzny
6	ELF

Tabela 3. *Tabela dopuszczalnych offsetów*

Offset	Miejsce dopasowania
*	Dowolne miejsce w pliku
n	n-ty bajt
EOF-n	n-ty bajt licząc od końca pliku
EP+n (ten i poniższe offsety działają tylko z plikami wykonywalnymi PE oraz ELF)	n-ty bajcie począwszy od Entry Point
EP-n	n-ty bajt przed Entry Point
Sx+n	n-ty bajt począwszy od początku sekcji o numerze x
Sx-n	n-ty bajt przed początkiem sekcji o numerze x
SL+n, SL-n	Jak wyżej, gdzie L odpowiada ostatniej sekcji w pliku

Sigtool

Ostatnie opisywane narzędzie przeznaczone jest przede wszystkim dla osób bezpośrednio związanych z opracowywaniem sygnatur wirusów oraz aktualizacją baz w projekcie ClamAV. *Sigtool* pozwala rozpakowywać, weryfikować oraz tworzyć cyfrowo podpisane bazy sygnatur, generować pliki różnicowe (*.cdiff), wydobywać kod makr z plików Office czy dokonywać normalizacji plików HTML, ułatwiając tworzenie właściwych sygnatur.

LibClamAV

Biblioteka *libclamav* jest elementem bezpośrednio odpowiedzialnym za wykrywanie wirusów. Jest ona bezpieczna dla wątków oraz udostępnia nieskomplikowane API (ang. *Application Programming Interface*) pozwalające programom na dostęp do uniwersalnych funkcji skanujących. Na Rysunku 1 przedstawiony został ogólny schemat działania biblioteki, a poniżej opisane poszczególne etapy. Artykuł koncentruje się na aktualnej wersji oprogramowania z repozytorium CVS, która wkrótce zastąpi obecną stabilną serię 0.8x.

Ustalanie typu pliku

Ze względów bezpieczeństwa oraz bezpośredniej pracy na deskryptorach, w celu ustalenia typu pliku *libclamav* analizuje jego zawartość, a nie nazwę w systemie plików. Coś, co z pozoru może wydawać się trywialne, okazuje się niekiedy twardym orzechem do zgryzienia. O ile rozpoznanie pliku wykonywalnego ELF nie stanowi problemu, o tyle stwierdzenie, czy dany plik jest archiwum tar wymaga już większego wysiłku. Ponieważ od wyniku tego procesu bezpośrednio zależy sukces skanowania, musi zostać on przeprowadzony w sposób możliwie najdokładniejszy. Biblioteka *libclamav* stosuje w tym celu trzy techniki:

- standardowe testy sprawdzające liczby magiczne (ang. *magic*

numbers), czyli unikatowe wartości wewnątrz pliku, charakterystyczne tylko dla danego typu,
- detekcję algorytmiczną opartą o analizę strukturalną lub zawartości pliku,
- skanowanie pliku pod kątem zawartości charakterystycznych fragmentów danych (na przykład fragmentów kodu HTML).

Wsparcie dla plików specjalnych

Gdy typ pliku zostanie ustalony, podejmowana jest decyzja dotycząca jego obsługi. Pliki specjalne, takie jak archiwa czy dokumenty wymagają wcześniejszego przetworzenia, polegającego na odpowiedniej ekstrakcji danych z ich wnętrza. Biblioteka posiada bezpośrednie wsparcie dla następujących typów:

- archiwa: zip, RAR, CHM (ang. *Compiled HTML*), cabinet, OLE2, tar, SIS (pakiety instalacyjne dla systemu Symbian),
- pliki skompresowane: gzip, bzip2, compress,
- pliki poczty: wszystkie popularne formaty uniksowe, TNEF (winmail.dat), PST,
- dokumenty: HTML, PDF, MS Office,
- pliki wykonywalne: PE (wraz z obsługą UPX, FSG, Petite, PESpin, WWPack32, Y0da Cryptor), ELF,
- inne: JPEG, GIF, RIFF (analiza plików pod kątem obecności exploitów).

Moduły obsługujące pliki specjalne zostały stworzone tak, aby wykrywać nieprawidłowości, które mogłyby zostać wykorzystane do wprowadzenia skanera w błąd. Na przykład, procedury obsługujące pliki zip są w stanie wykryć manipulacje w lokalnych nagłówkach archiwów, fałszywe oraz ukryte wpisy. Dodatkowym elementem ochrony są limity związane z poziomem rekursji, liczbą oraz rozmiarem rozpakowywanych plików, które stanowią zabezpieczenie przed atakami typu *Denial of Service*, wykorzystującymi bomby archiwowe (niewielkie archiwa zawierające dużą liczbę wysoce skompresowanych plików).

Listing 2a. *Przykład użycia biblioteki libclamav*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <clamav.h> /* plik nagłówkowy biblioteki libclamav */
int main(int argc, char **argv)
{
        int fd, ret;
        unsigned long int size = 0;
        unsigned int sigs = 0;
        long double mb;
        const char *virname;
        struct cl_engine *engine = NULL;
        struct cl_limits limits;
    if(argc != 2) {
        printf("Usage: %s file\n", argv[0]);
        exit(2);
    }
    if((fd = open(argv[1], O_RDONLY)) == -1) {
        printf("Can't open file %s\n", argv[1]);
        exit(2);
    }
    /* wczytaj wszystkie bazy z domyślnego katalogu */
    if((ret = cl_load(cl_retdbdir(), &engine, &sigs, CL_DB_STDOPT))) {
        printf("cl_load: %s\n", cl_strerror(ret));
        close(fd);
        exit(2);
    }
    printf("Loaded %d signatures.\n", sigs);
    /* skompiluj silnik */
    if((ret = cl_build(engine))) {
        printf("Engine compilation error: %s\n", cl_strerror(ret));;
        cl_free(engine);
        close(fd);
        exit(2);
    }
    /* ustaw limity archiwów */
    memset(&limits, 0, sizeof(struct cl_limits));
    limits.maxfiles = 1000; /* maksymalna liczba plików, które zostaną
                    rozpakowane */
    limits.maxfilesize = 10 * 1048576; /* maksymalny rozmiar zarchiwizowanego
                    pliku */
    limits.maxreclevel = 5; /* maksymalny poziom rekursji */
    limits.maxratio = 200; /* maksymalny stosunek rozmiaru pliku
                    skompresowanego do oryginału */
    /* przeskanuj wskazany deskryptor */
    if((ret = cl_scandesc(fd, &virname, &size, engine, &limits, CL_SCAN_
                    STDOPT)) == CL_VIRUS) {
        printf("Virus detected: %s\n", virname);
    } else {
        if(ret == CL_CLEAN) {
            printf("No virus detected.\n");
        } else {
            printf("Error: %s\n", cl_strerror(ret));
            cl_free(engine);
            close(fd);
            exit(2);
        }
    }
    close(fd);
```

Zastosowanie znajdują także dodatkowe informacje umieszczone w archiwach (metadane), które *libclamav* pozwala skanować za pomocą specjalnych sygnatur, umożliwiając w niektórych przypadkach wykrycie złośliwego oprogramowania wewnątrz zaszyfrowanych archiwów.

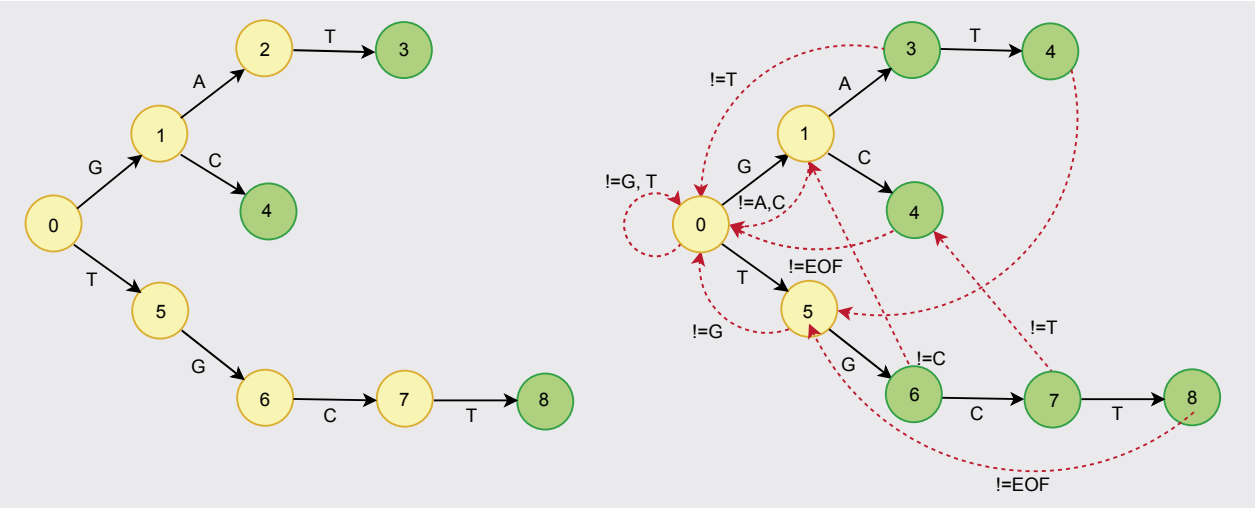
Skanowanie sygnaturami

Aby efektywnie skanować pliki w poszukiwaniu dziesiątek tysięcy wirusów, *libclamav* stosuje dwa algorytmy: Aho-Corasick oraz rozszerzony algorytm Boyer-Moore'a. Są to algorytmy dopasowania wielowzorcowego, umożliwiające przeszukiwanie plików pod kątem wielu sygnatur w tym samym czasie.

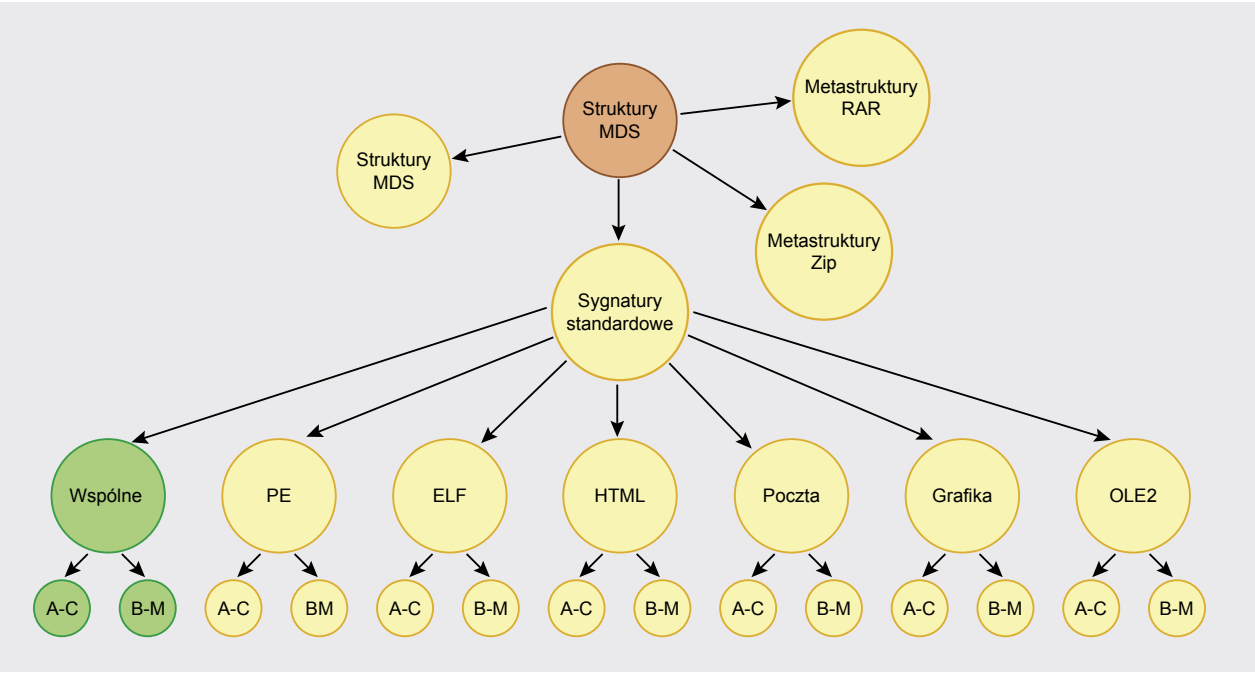
Dodatkowo, biblioteka obsługuje akceleratory sprzętowe znacznie przyspieszające skanowanie.

Algorytm Aho-Corasick na podstawie zbioru wzorców (u nas sygnatur wirusów) buduje specjalne drzewo, które następnie zostaje przekształcone w automat skończony.

Na Rysunku 2 przedstawiony został przykład drzewa oraz odpowiadającego mu automatu, stworzonych dla zbioru wzorców P = {GAT, GC, TGCT}. Stany automatu zaznaczone kolorem zielonym, to stany końcowe – gdy automat znajdzie się w takim stanie, oznaczać to będzie poprawne dopasowanie jednego ze wzorców. Czarne strzały (krawędzie drzewa) reprezentują funkcję przejścia i umożliwiają zmianę stanu, gdy znak wejściowy jest znakiem przez nie akceptowanym. Czerwone strzały reprezentują natomiast funkcję porażki i pozwalają na zmianę stanu, gdy niemożliwe jest to poprzez funkcję przejścia. Automat rozpoczyna działanie w stanie zerowym (który jest korzeniem drzewa i stanem początkowym automatu) i dokonuje odpowiednich zmian stanu



Rysunek 2. *Przykład drzewa i automatu Aho-Corasick*



Rysunek 3. *Drzewo sygnatur w pamięci*

dla każdego znaku tekstu wejściowego. W omawianym przykładzie, przeszukiwanie tekstu odbywa się w czasie liniowym – każdy ze znaków tekstu wejściowego jest sprawdzany tylko jeden raz. Implementacja algorytmu w ClamAV nie zapewnia takiego czasu działania automatu, ponieważ w celu ograniczenia zużycia pamięci wprowadzone zostały ograniczenia dla głębokości drzewa A-C, jednak w dalszym ciągu wydajność jest zadowalająca. Dodatkowo, algorytm został rozszerzony o obsługę podstawowych znaków zastępczych (ang. *wild cards*) opisanych w następnym punkcie.

Rozszerzony algorytm Boyer-Moore'a podobnie do wersji oryginalnej wykorzystuje ideę tablicy przesunięć (ang. *shift table*), umożliwiającej pominięcie części tekstu wejściowego, w ktorym nie może nastąpić dopasowanie. Zasadnicza część algorytmu pracuje nie na pojedynczych znakach, lecz na haszach obliczanych z trzech kolejnych znaków, które służą jako indeksy tablicy przesunięć oraz specjalnej tablicy sufiksów. Z tego powodu algorytm jest stosowany tylko dla sygnatur statycznych (nie zawierających znaków zastępczych).

Proces skanowania sygnaturami może zostać wielokrotnie przyspieszony przez zastosowanie akceleratora sprzętowego. ClamAV posiada wsparcie dla platformy NodalCore firmy Sensory Networks, w ramach której dostępne są karty PCI o przepustowościach do 2 Gb/s. Ich wykorzystanie nie tylko przyspiesza samo skanowanie, ale także w znaczny sposób redukuje obciążenie CPU, co ma niebagatelne znaczenie dla systemów filtrujących setki tysięcy bądź miliony plików dziennie. Informacje techniczne oraz ceny kart można znaleźć na stronach producenta.

Typy sygnatur wirusów

Libclamav obsługuje kilka typów sygnatur, pozwalających na opis złośliwego oprogramowania na różne sposoby.

Głównym elementem sygnatur standardowych jest fragment kodu, który w jednoznaczny sposób identyfikuje cel. Aby uniknąć fałszywych alarmów oraz umożliwić przechowywanie wpisów w plikach tekstowych, jest on zakodowany do postaci napisu złożonego z liczb szesnastkowych, może także zawierać znaki zastępcze opisane w Tabeli 1, które pozwalają zwiększyć elastyczność sygnatury. Sygnatury standardowe mogą występować w dwóch postaciach: uproszczonej oraz pozwalającej na dokładniejszy opis celu. W pierwszym przypadku są to sygnatury umieszczane w bazach **.db*, o następującym formacie:

```
NazwaWirusa=SygnaturaSzesnastkowa
```

Sygnatury w postaci rozszerzonej umieszczane są w plikach *.ndb i wyglądają następująco:

```
NazwaWirusa:KodCelu:Offset:
SygnaturaSzesnastkowa[:
MinimalnaWersjaSilnika:[MaksymalnaWS]]
```

Pozwalają one na wskazanie typu pliku oraz miejsca, w którym powinno nastąpić dopasowanie sygnatury. W Tabeli 2 oraz 3 przedstawione zostały możliwe wartości dla drugiego oraz trzeciego pola.

W przypadku złośliwego oprogramowania, które w żaden sposób nie modyfikuje swoich plików, takiego jak większość dialerów czy prostych koni trojańskich, możliwe jest użycie sygnatur wykorzystujących sumy MD5. Umieszczane są one w plikach **.hdb* i mają następujący format:

```
MD5:RozmiarPliku:Nazwa
```

Dzięki nim, użytkownicy mogą w łatwy sposób tworzyć swoje własne sygnatury – wystarczy wykorzystać opcję *--md5* programu *sigtool*:

```
sigtool --md5 evil.exe > evil.hdb
```

a następnie przenieść plik *evil.hdb* do katalogu z bazami ClamAV.

Ostatni typ sygnatur przeznaczony jest do opisu plików wewnątrz archiwów i w tym celu wykorzystuje wspomniane wcześniej metadane zawarte w archiwach. Ma on zastosowanie przede wszystkim w przypadku robaków internetowych używających zaszyfrowanych archiwów w charakterze nośników, a których pliki zachowują jakąś charakterystyczną własność, taką jak nazwa, rozmiar czy suma kontrolna. Sygnatury te są przechowywane w plikach *.zmd* oraz *.rmd*, odpowiednio dla archiwów zip i RAR, w następującym formacie:

```
NazwaRobaka:zaszyfrowany(0,1):nazwa
pliku:oryginalny rozmiar:rozmiar po
kompresji:crc32:metoda kompresji:numer
porządkowy pliku w archiwum:maksymalna
liczba zagnieżdżonych archiwów
```

Jeżeli pewne parametry ulegają zmianie, można użyć gwiazdki w celu zignorowania ich wartości. Przykładami robaków pomyślnie wykrywanych tą specyficzną metodą są między innymi Worm.Padowor.A, Worm.Kimazo.A oraz warianty robaka Bagle.

W celu optymalnego wykorzystania, w trakcie ładowania baz sygnatury dzielone są na odpowiednie grupy, tworząc drzewo przedstawione na Rysunku 3. Dane skanowane są w pierwszej kolejności sygnaturami zgodnymi z ich typem, następnie sygnaturami wspólnymi oraz MD5. Sygnatury wykorzystujące metadane sprawdzane są w momencie rozpakowywania archiwów.

Pliki skanowane są w pierwszej kolejności sygnaturami zgodnymi z ich typem, następnie sygnaturami wspólnymi oraz MD5.

Listing 2b. *Przykład użycia biblioteki libclamav*

```
    /* oblicz przybliżony rozmiar przeskanowanych danych  */
    mb = size * (CL_COUNT_PRECISION / 1024) / 1024.0;
    printf("Size of scanned data: %2.2Lf MB\n", mb);
    /* zwolnij pamięć zajmowaną przez silnik */
    cl_free(engine);
    exit(ret == CL_VIRUS ? 1 : 0);
}
```

Pliki CVD

Podstawowym nośnikiem dla sygnatur w ClamAV są pliki CVD, które w istocie są skompresowanymi oraz cyfrowo podpisanymi archiwami tar. Dystrybuowane są dwa pliki: *daily.cvd* oraz *main.cvd*. Pierwszy z nich to mniejsza baza służąca do codziennych aktualizacji, drugi to z kolei główne archiwum sygnatur, do którego średnio co półtorej miesiąca przenoszona jest część wpisów z *daily.cvd*. Bazy zawierają pliki tekstowe z sygnaturami w opisanych wyżej formatach. W celu zmniejszenia obciążenia mirrorów opracowany został system aktualizacji różnicowych, który znosi potrzebę ściągania kompletnych plików z bazami. Zamiast tego, *freshclam* pobiera tylko specjalny skrypt z ostatnio wprowadzonymi zmianami i dokonuje odpowiednich aktualizacji lokalnych plików.

Algorytmiczna detekcja wirusów

Niektóre zaawansowane wirusy wymagają specjalnego podejścia. Dotyczy to przede wszystkim wirusów wysoce polimorficznych, które uniemożliwiają bezpośrednie wykrycie za pomocą sygnatur. Część skanerów antywirusowych w celu ich wykrycia stara się wykorzystywać uniwersalne podejście oparte o emulację kodu, jednak w przypadku nowoczesnych wirusów wykorzystujących zaawansowane sztuczki bądź coraz to nowe rozszerzenia procesorów, może się ono często okazać nieefektywne. ClamAV stawia na precyzyjną detekcję opartą o dedykowane algorytmy. Metoda ta jest bardziej pracochłonna, jednak pozwala na dokładne oraz co bardzo ważne – szybkie wykrycie wirusa przez skaner. W chwili obecnej, wszystkie algorytmy muszą być umieszczane bezpośrednio w źródłach programu, jednak już wkrótce możliwa będzie ich dystrybucja w postaci bajtkodu (ang. bytecode) obok standardowych sygnatur. Na Listingu 1 przedstawiony został krótki fragment kodu odpowiedzialny za wykrywanie wirusa W32.Parite.B, Czytelnicy zainteresowani bardziej skomplikowanymi przykładami zachęcani są do lektury źródeł.

Przykład użycia biblioteki

Biblioteka *libclamav* pozwala na łatwą rozbudowę istniejącego oprogramowania o skanowanie antywirusowe. Na Listingu 2 przedstawiony został kompletny przykład wykorzystania biblioteki – proste narzędzie umożliwiające skanowanie pojedynczych plików z linii poleceń. Wykorzystane zostały w nim podstawowe funkcje: `cl_load()` wczytuje pojedynczą bazę lub wszystkie bazy ze wskazanego katalogu, `cl_retdbdir()` zwraca ścieżkę domyślnego katalogu z bazami, `cl_build()` dokonuje kompilacji silnika, `cl_scandesc()` skanuje wskazany deskryptor, `cl_strerror()` tłumaczy kod błędu na komunikat w języku angielskim i w końcu `cl_free()` zwalnia pamięć zajmowaną przez silnik. Program zwraca do systemu kod 1 w przypadku wykrycia infekcji, 0 w przypadku jej nie wykrycia oraz 2, w przypadku wystąpienia błędu.

Infrastruktura sieciowa

Projekt ClamAV posiada zaawansowaną, rozbudowywaną od kilku lat infrastrukturę serwerów lustrzanych (mirrorów). Jest ona sponsorowana przez firmy, organizacje oraz jednostki edukacyjne, które udostępniają szerokopasmowe łącza internetowe dla potrzeb dystrybucji baz z sygnaturami wirusów. Ponad sto dwadzieścia maszyn w blisko czterdziestu krajach zapewnia szybką oraz bezawaryjną aktualizację baz. W celu optymalnego dostępu do aktualizacji, dostęp do serwerów następuje poprzez następujące adresy:

- db.XY.clamav.net – wskazuje na mirrory dostępne w kraju o kodzie XY,
- db.local.clamav.net – próbuje przekierować klienta do najbliższej puli mirrorów poprzez sprawdzenie jego adresu w bazie GeoIP,
- database.clamav.net – rekord round-robin wskazujący na zbiór najszybszych i najbardziej niezawodnych mirrorów.

W przypadku Polski, zalecana konfiguracja polega na umieszczeniu następujących dwóch wpisów w pliku *freshclam.conf*:

```
DatabaseMirror db.pl.clamav.net
DatabaseMirror database.clamav.net
```

W przypadku gdy połączenie z mirrorami krajowymi nie powiedzie się, *freshclam* automatycznie użyje drugiego wpisu.

Możliwe zastosowania

Mimo, że ClamAV został zaprojektowany przede wszystkim z myślą o skanowaniu poczty, pomyślnie sprawdza się także w innych zastosowaniach. Na stronie głównej projektu, w dziale *Third-party software* znaleźć można listę ponad stu programów współpracujących z ClamAV, pozwalających na wykorzystanie go do filtrowania poczty na poziomie protokołów SMTP, POP3 oraz czytników poczty, skanowania systemu plików, strumieni HTTP, zasobów FTP i do wielu, wielu innych celów.

Podsumowanie

Skaner antywirusowy powinien być jednym z podstawowych narzędzi w arsenale każdego administratora sieci. Clam AntiVirus to elastyczne oprogramowanie, dające szerokie możliwości konfiguracji oraz zastosowania, a dzięki swojej licencji i szybkim reakcjom na nowe zagrożenia, obala mit (oraz częstą wymówkę lekkodusznych administratorów) o niebotycznych kosztach ochrony antywirusowej. ●

W Sieci

- *http://www.clamav.net/* – strona domowa projektu Clam AntiVirus,
- *http://www.dazuko.org/* – strona domowa projektu Dazuko,
- *http://www.sensorynetworks.com/* – strona producenta akceleratorów sprzętowych.

O autorze

Autor jest twórcą oraz liderem projektu Clam AntiVirus. Absolwent informatyki na Uniwersytecie Mikołaja Kopernika w Toruniu, wieloletni entuzjasta oprogramowania Open Source oraz żółwi wodnych.