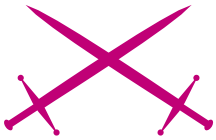


Przecinanie kabla – atak TCP Reset

Marcin Ulikowski

stopień trudności

Protokół TCP został stworzony prawie 30 lat temu, gdy duże liczby naturalne wydawały się większe niż dzisiaj. Twórcy protokołu, podobnie jak w przypadku kończącej się 32-bitowej przestrzeni adresowej IP, nie przewidzieli, że taki sam rozmiar pewnego pola w nagłówku TCP może stać się po wielu latach problemem.

Zagrożenia związane z możliwością przechwycenia sesji TCP (*Transmission Control Protocol*) były znane specjalistom bezpieczeństwa sieciowego od wielu lat. Oryginalna specyfikacja protokołu TCP zawiera rozwiązanie umożliwiające poprawną transmisję pakietów odbieranych w losowej kolejności oraz jednoczesne odrzucanie duplikatów. Dodatkowo chroni transmisję przed atakiem wstrzyknięcia danych (ang. *injection*). Numery sekwencyjne zawarte w każdym nagłówku TCP przenoszą informacje o kolejności, w jakiej otrzymywane dane powinny być ze sobą łączone. Jednocześnie mają zapewniać bezpieczeństwo strumienia danych TCP. Chociaż świadomość związanego z nimi potencjalnego zagrożenia istniała od wielu lat, niewiele zrobiono, aby je wyeliminować i jeszcze mniej na ten temat napisano.

Ataki oparte na numerach sekwencyjnych TCP wykorzystywały słabości w generowaniu pseudolosowych numerów, którymi posługuje się podczas nawiązywania połączenia. Wczesne implementacje stosów TCP w przypadku większości systemów nie mogły pochwalić się odpowiednio dobrą losowością numerów sekwencyjnych. Fakt ten umożliwiał odgadnięcie inicjującego numeru sekwencyjnego (*Initial Sequence Number*) w bardzo krótkim czasie. Znanym przykładem wykorzystania słabej losowości numerów sekwencyjnych był atak przeprowadzony na komputer Tsutomu Shimomury przez Kevina Mitnicka w 1994 roku.

Z biegiem czasu algorytmy generujące losowe numery stawały się coraz lepsze i wiodące systemy operacyjne (nawet Windows,

Z artykułu dowiesz się...

- z czego wynika słabość protokołu TCP,
- w jaki sposób przeprowadzić przykładowy atak TCP Reset,
- jak zabezpieczyć system przed atakiem tego rodzaju.

Co powinieneś wiedzieć...

- podstawowe informacje na temat protokołów sieciowych i komunikacji w internecie,
- znajomość języka ANSI C będzie pomocna, ale nie niezbędna.

który wcześniej nie znał znaczenia słowa „losowy") mogły pochwalić się znacznie większym poziomem bezpieczeństwa. Równocześnie z lepszymi algorytmami losującymi zwiększała się także przepustowość Internetu. Szybkie łącza stały się tańsze i bardziej dostępne. Problem numerów sekwencyjnych powrócił.

Nagłówek TCP

Protokół TCP należy do warstwy transportowej modelu OSI i jest odpowiedzialny za kontrolę przepływu poprawności danych. Podstawowy mechanizm protokołu zakłada konieczność potwierdzenia przez odbiorcę wszystkich otrzymanych segmentów TCP. Nadawca wysyła pewną porcję danych. Gdy nadejdzie potwierdzenie od odbiorcy, nadawca może wysłać następną porcję danych. Gdy odbiorca sygnalizuje błędy lub nie wyśle potwierdzenia, następuje retransmisja wszystkich danych, począwszy od pierwszego niepotwierdzonego segmentu. Analizując nagłówek TCP skupimy się tylko na tych elementach, których znajomość jest kluczowa do zrozumienia idei działania ataku TCP Reset.

Numer sekwencyjny

Gdy strumień danych zostaje podzielony na pakiety, mogą one dotrzeć do odbiorcy w nieprawidłowej kolejności, na przykład wskutek przeciążenia sieci. Numer sekwencyjny o długości 32 bitów pełni rolę identyfikatora dla każdego wysyłanego pakietu, dzięki czemu strumień danych może zostać poprawnie złożony w całość po drugiej stronie. Identyfikatory pakietów nie mogą zaczynać się od liczby 0 ani żadnej innej stałej wartości. Musi to być możliwie najlepsza losowa liczba, aby umożliwić przeprowadzenie poprawnej transmisji, która nie zostanie zakłócona.

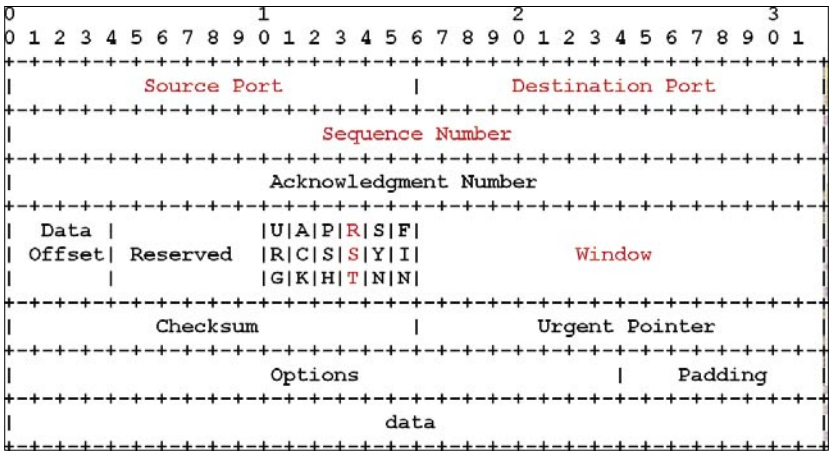
Okno

Rozmiar okna (ang. *window*) odzwierciedla maksymalną liczbę pakietów (oraz buforowanych danych), które mogą być wysłane bez konieczności oczekiwania na pozytywną odpowiedź. Duże okna TCP poprawiają wydajność protokołu TCP/IP podczas przesyłania dużej ilości danych między nadawcą i odbiorcą. Zbyt duży rozmiar okna powoduje większe zużycie pamięci i dłuższą ponowną retransmisję w razie utraty pakietów. Natomiast zbyt mały rozmiar okna obniża szybkość transmisji wydłużając czas potrzebny na oczekiwanie potwierdzenia. W typowym połączeniu TCP maksymalny rozmiar okna jest ustalany na początku połączenia i jest ograniczony do 64kB, co wynika z 16-bitowej długości tego pola. Po otrzymaniu wszystkich pakietów mieszczących się w oknie, host może odesłać pakiet z zerowym rozmiarem okna, informując drugą stronę połączenia, aby wstrzymała na moment wysyłanie kolejnych pakietów.

Każda implementacja stosu TCP dla systemów operacyjnych czy urządzeń sieciowych ma zdefiniowaną własną wielkość okna ustalaną podczas nawiązywania połączenia. Rozmiar może się zmieniać dynamicznie podczas transmisji (na przykład zmniejszyć jeśli duża część pakietów nie dociera do odbiorcy). Bardzo pomocna będzie Tabela 1, która zawiera początkowe rozmiary okna dla popularnych systemów.

Flagi

Nagłówek pakietu TCP może zawierać sześć bitów kontrolnych (SYN,



Rysunek 1. *Nagłówek TCP, nazwy omawianych pól mają kolor czerwony*

Tabela 1. *Początkowy rozmiar okna w zależności od systemu operacyjnego*

System operacyjny	Początkowy rozmiar okna
Linux 2.4 - 2.6	5840
Linux 2.2	16384, 32768
Linux 2.0.3x	512, 16384
Windows XP	16384, 64240
Windows 2000	16384, 64512
Windows 2003	65535
Windows 9x, NT 4	8192
FreeBSD 5x, 6x	65535

Jeszcze większe okna

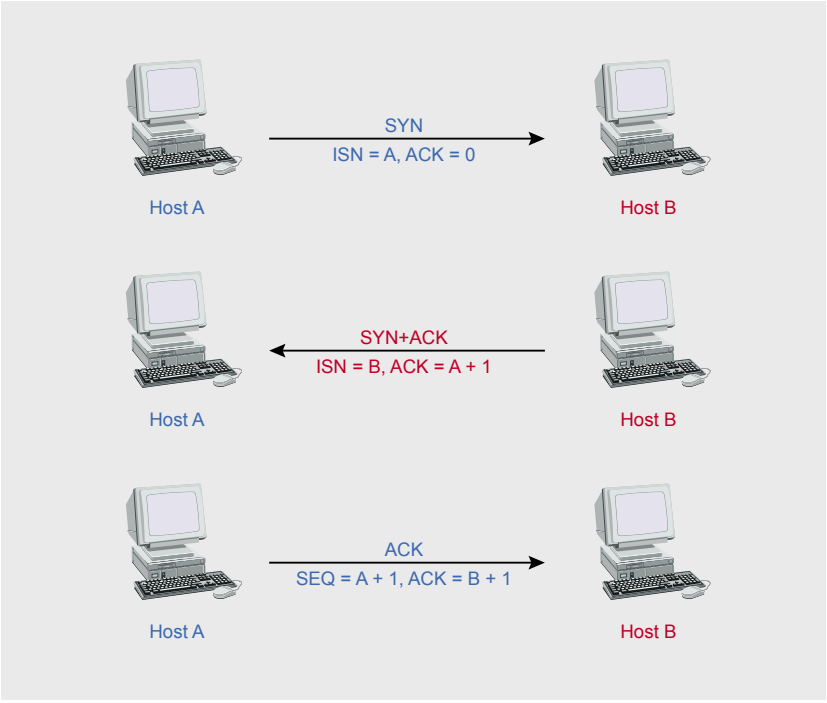
Dokument RFC-793 z 1981 roku określa maksymalny rozmiar okna TCP jako 64kB. Jest to mało, jak na dzisiejsze łącza. Dlatego nowszy dokument RFC-1323 z 1992 roku wprowadza rozszerzenie do protokołu o nazwie *Window Scaling*, które umożliwia zwiększenie rozmiaru okna nawet do 1GB. Wymaga to umieszczenia dodatkowej opcji pod nagłówkiem TCP. Większe okno oznacza zwiększoną szybkość transmisji na łączach o dużej przepustowości.

ACK, PSH, URG, RST oraz FIN) zwanych flagami. Określają one, co zawiera pakiet i jaką spełnia funkcję podczas transmisji. Z naszego punktu widzenia najbardziej interesujące są trzy z nich: SYN, ACK i RST. Flaga SYN (*synchronize*) jest wysyłana razem z pakietem nawiązującym połączenie. Dołączany jest do niej losowy numer sekwencyjny (ISN). Druga strona odsyła własny pakiet z ustawionymi flagami SYN+ACK. Wysyła także własny losowy numer sekwencyjny (ISN) oraz numer potwierdzenia (*acknowledgment*), który odpowiada otrzymanemu wcześniej ISN zwiększonemu o jeden. Jeśli strona próbująca nawiązać połączenie otrzyma pakiet i dokona pozytywnej weryfikacji numeru potwierdzenia (polegającej na sprawdzeniu czy otrzymany numer potwierdzenia jest równy wysłanemu wcześniej ISN plus jeden), może wysłać pakiet z flagą ACK. Jeśli numer potwierdzenia w trzecim pakiecie jest równy ISN+1 z pakietu poprzedniego to oznacza, że połączenie zostało nawiązane. Teraz dane mogą być wysyłane w obu kierunkach. Każdy kolejny pakiet będzie miał zwiększony numer sekwencyjny. Aby natychmiast zerwać połączenie wystarczy wysłać pakiet z ustawioną flagą RST (*reset*) lub RST+ACK, jeśli chcemy otrzymać potwierdzenie.

Porty

Nagłówek TCP zawiera dwa 16-bitowe pola w których występują informacje na temat portu źródłowego i portu docelowego. Komputery podłączone do sieci komunikują się ze światem zewnętrznym przez porty, które pozwalają zarówno na wysyłanie, jak i odbieranie danych. Porty umożliwiają identyfikację konkretnego połączenia oraz pozwalają ustalić, które pakiety przesyłane są w jego zakresie. Z tego powodu dwa połączenia wychodzące nie mogą wykorzystywać tego samego portu źródłowego.

Kiedy chcemy nawiązać połączenie z konkretną usługą na innym komputerze, na przykład SSH, numer portu docelowego w pakiecie SYN będzie miał wartość 22 (jest to domyślny numer portu dla tej usługi, chociaż może być inny). Zwykle mamy możliwość wyboru portu, na jaki chcemy się połączyć, ponieważ aplikacja udostępnia taką możliwość. Port źródłowy ustalany jest przez system operacyjny (aczkolwiek możliwe jest, aby wyboru dokonała aplikacja). Zwykle jest to kolejny wolny numer. Nie jest on wybierany, jak mogłoby się wydawać, z pełnej puli 65536 portów. Część z nich jest zarezerwowana dla uprzywilejowanych procesów (najczęściej pierwsze 1024) oraz dla funkcji



Rysunek 2. *Nawiązywanie połączenia TCP, tzw. three-way handshake*

Tabela 2. *Numery pierwszych dostępnych dla połączeń portów źródłowych*

System operacyjny	Pierwszy dostępny port TCP
Linux 2.4.x	1024
Windows XP (SP1, SP2)	1050
Windows 2000 SP3	1060
Windows 2000 SP4	1038
*BSD	1024

Losowe porty

Większość systemów operacyjnych i urządzeń sieciowych wybierając port źródłowy dla tworzonego połączenia korzysta z kolejnego wolnego. Wyjątkiem jest OpenBSD, który losuje porty ze zbioru nieużywanych przez inne połączenia. Twórcy tego systemu bardzo duży nacisk kładą na jego bezpieczeństwo. W przypadku systemu Linux, podobną funkcję oferuje łatka grsecurity.

Atak z flagą SYN

Zamiast flagi RST możemy z powodzeniem wykorzystać flagę SYN do przerwania połączenia. Większość implementacji stosu TCP odpowie wysyłając pakiet z ustawioną flagą RST oraz właściwym numerem sekwencyjnym, powodując zakończenie transmisji. W ten sposób prowokujemy jedną ze stron do przerwania połączenia.

systemowych jak na przykład translacji adresów (w tym przypadku od 49152 do 65535).

Podobnie jak w przypadku początkowego rozmiaru okna, poszczególne implementacje stosu TCP różnią się pod względem ilości portów zarezerwowanych dla uprzywilejowanych procesów. Tabela 2 przedstawia numery pierwszych dostępnych do użytku portów w przypadku popularnych systemów operacyjnych.

Przerywanie połączeń

Atak TCP Reset polega na zmuszeniu jednej ze stron do zerwania połączenia podszywając się pod adres drugiego komputera. Wyobraźmy sobie, że komputer A ma nawiązane połączenie z komputerem B. Trzeci komputer C chce zerwać to połączenie. Wysyła w tym tym celu do komputera A pakiet z ustawioną flagą RST oraz z adresem źródłowym należącym do komputera B. Pakiet zawiera także porty - źródłowy i docelowy odpowiadające atakowanemu połączeniu oraz aktualnie obowiązujący (czyli taki, który spodziewa się otrzymać komputer A) numer sekwencyjny. Gdy komputer A otrzyma ten pakiet, natychmiast zerwie połączenie.

Głównym problemem jest odgadnięcie numeru sekwencyjnego, który zostanie zaakceptowany przez jedną ze stron połączenia. Ponieważ numery sekwencyjne mają długość 32 bitów, liczba możliwych permutacji wynosi 4294967296, czyli ponad 4 miliardy. Jeśli komputer C miałby zgadywać numer sekwencyjny to jego szanse powodzenia są znacznie mniejsze niż szanse na główną wygraną w popularnej grze liczbowej.

Atakujący może także wykorzystać wszystkie możliwości, jednak w najgorszym przypadku będzie to wymagało wysłania 160GB danych, przy założeniu, że każdy wysłany pakiet RST ma długość 40 bajtów. Czas potrzebny na wysłanie takiej ilości danych przy użyciu łącza DSL 1Mb/s wynosi około 2 tygodni. Połączenie między komputerem A i B zostanie zakończone w krótszym czasie w sposób naturalny. Rozwiązaniem problemu jest rozmiar okna w pakiecie TCP. Ponieważ rozmiar okna oznacza maksymalną ilość pakietów, na które jedna ze stron nie musi odsyłać potwierdzenia, wystarczy wysłać pakiet ze zbliżoną wartością numeru sekwencyjnego (w zależności od rozmiaru okna). System musi zaakceptować pakiet z numerem sekwencyjnym różniącym się w granicy rozmiaru okna, ponieważ wie, że dane mogą docierać w nieodpowiedniej kolejności.

Na przykład, oczekiwanym numerem sekwencyjnym jest SEQ i rozmiar okna po stronie komputera A wynosi 16384 bajtów. Teraz wystarczy, że atakujący komputer C wyśle do A pakiet z numerem sekwencyjnym z przedziału od SEQ do SEQ+16384-1 (przy założeniu, że bufor dla okna komputera A jest pusty, to znaczy nie otrzymał jeszcze danych), aby został zaakceptowany.

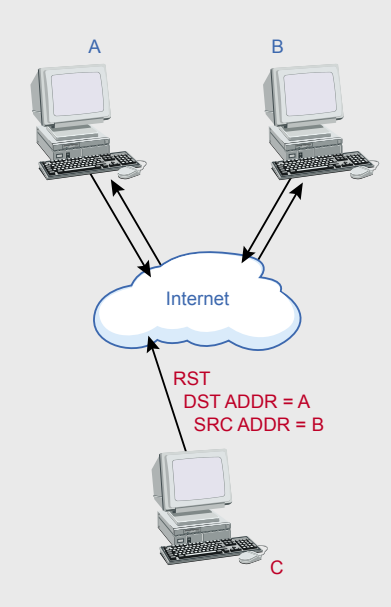
Oznacza to, że łączna ilość danych, które musimy wysłać maleje do 10MB, ponieważ ilość pakietów wynosi teraz 262144. W przypadku większego rozmiaru okna, ilość wysłanych danych będzie jeszcze mniejsza. Zależność przedstawia Tabela 3. Warto zauważyć, że podane wartości odpowiadają najgorszemu scenariuszowi. W rzeczywistości ilość potrzebnego czasu będzie średnio o połowę mniejsza. Ponadto, jeśli wykorzystywane będzie rozszerzenie protokołu TCP *Window Scaling*, ilość potrzebnych pakietów zmniejszy się jeszcze bardziej. Teoretycznie, w przypadku największego możliwego rozmiaru okna będą to tylko 4 pakiety.

Przykładowy atak

Do przeprowadzenia własnego ataku wykorzystamy gotowy program *tcprst.c*, którego główna zaleta to fakt, iż nie potrzebuje on żadnych zewnętrznych bibliotek. Wykorzystuje gniazda typu RAW do wysyłania pakietów. Do poprawnego działania programu wymagane są najwyższe uprawnienia oraz odpowiedni sposób kompilacji:

```
# gcc -DNOSCRIPTKID -o tcprst tcprst.c
```

Nasze narzędzie jest gotowe do pracy. Teraz musimy utworzyć przykładowe połączenie, które następnie przerwiemy. Do tego celu wykorzystamy dwa komputery połączone w sieci lokalnej z dostępem do Internetu. Pierwszy o adresie 192.168.0.101 działa pod kontrolą Linuksa.



Rysunek 2. *Schemat ataku, w którym komputer C przerywa połączenie między komputerami A i B*

Tabela 3. *Czas ataku w zależności od wielkości okna TCP przy użyciu łącza 1Mb/s*

Rozmiar okna	Ilość danych [MB]	Potrzebny czas [s]
4096	40	320
8192	20	160
16384	10	80
32768	5	40
65535	2,5	20

```
tcprst.c - DoS over TCP long-time connections
(c) Marcin Ulikowski <elceef@itsec.pl>
64.223.167.99:80 -> 192.168.0.1:1573 (win=64000)
RST counter: 8192  ISN guess: 524288000
RST counter: 16384  ISN guess: 1048576000
RST counter: 24576  ISN guess: 1572864000
RST counter: 32768  ISN guess: 2097152000
RST counter: 40960  ISN guess: 2621440000
RST counter: 49152  ISN guess: 3145728000
RST counter: 57344  ISN guess: 3670016000
RST counter: 65536  ISN guess: 4194304000
Total data sent: 2621KB
(root@osiris ~)#
```

Rysunek 3. *Program tcprst podczas pracy*

Wykorzystamy go do przeprowadzenia ataku. Drugi komputer o adresie 192.168.0.1 działa pod kontrolą systemu Windows XP. Za jego pomocą nawiążemy długotrwałe połączenie z usługą HTTP na porcie 80:

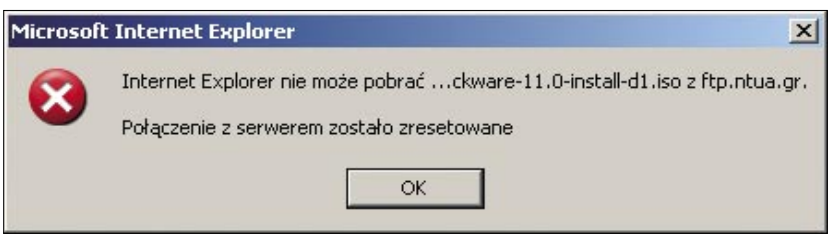
```
C:\>netstat -n |find ":80"
TCP    192.168.0.1:1573
64.223.167.99:80
USTANOWIONO
```

Gdy znamy już parametry połączenia, czyli adresy i porty, musimy zdecydować, którą stronę wykorzystać do przerwania komunikacji. Zależy nam na czasie potrzebnym do przeprowadzenia ataku, który będzie się różnił w zależności od adresu wybranego komputera. Czas ataku zależy od przepustowości łącza między nami (atakującym), a atakowanym komputerem oraz rozmiaru okna (czyli także od systemu operacyjnego) użytym przez komputer, do którego będziemy wysyłać pakiety. My wykorzystamy komputer z naszej sieci lokalnej, ponieważ szybkość połączenia będzie wielokrotnie większa niż w przypadku komputera `64.223.167.99` oraz dlatego, że działa pod kontrolą systemu z większym rozmiarem okna (mniejsza liczba pakietów do wysłania). Do identyfikacji systemu operacyjnego (w celu poznania początkowego rozmiaru okna) możemy użyć skanera portów *nmap* z opcją `-O`.

Kiedy już wybraliśmy adres komputera, pozostaje nam uruchomienie *tcprst*. Program wymaga czterech parametrów: adresu komputera pod który się podszywamy (czyli adres źródłowy, opcja `-S`), portu źródłowego z którego korzysta ten komputer (parametr `-s 80`) oraz adresu i portu na który zostaną wysłane pakiety RST. Istnieje jeszcze możliwość określenia rozmiaru okna przy pomocy dodatkowej, piątej opcji `-w` (domyślny rozmiar to 16384). Ponieważ atakowany komputer działa pod kontrolą Windows XP, zgodnie z Tabelą 1 zdefiniujemy rozmiar na 64000 bajtów:

```
# ./tcprst -S 64.223.167.99 -s 80 -D
192.168.0.1 -d 1573 -w 64000
```

Program informuje nas o postępach wyświetlając licznik pakietów oraz wysłane numery sekwencyjne. Po zakończeniu lub jeszcze w trakcie pracy na atakowanym komputerze powinien wyświetlić się komunikat podobny do tego pokazanego na Rysunku 4. Dzięki dużej przepustowości sieci lokalnej, wszystkie pakiety zostały wysłane w ciągu niecałych trzech sekund.



Rysunek 4. *Komunikat potwierdzający, że atak przebiegł pomyślnie*

Fakt, że przeprowadzony atak zakończył się powodzeniem potwierdza także informacja pochodząca z programu *netstat* na atakowanym systemie:

```
C:\>netstat -n |find ":80"
TCP    192.168.0.1:1573
64.223.167.99:80
CZAS_OCZEKIWANIA
```

Zachęcam do zapoznania się z kodem źródłowym naszego narzędzia. Jest krótki i przejrzysty, co ułatwi jego analizę i pozwoli zapoznać się z atakiem od strony programowej.

Kto jest zagrożony

Ponieważ jest to wada protokołu TCP, czyli kręgosłupa Internetu, zagrożeni są wszyscy użytkownicy, chociaż nie bezpośrednio. Szczególnie narażone są długotrwałe połączenia, jak na przykład sesje TCP wykorzystywane przez protokół BGP (*Border Gateway Protocol*). Jego zadaniem jest wykonywanie routingu w sieciach pracujących z protokołem TCP/IP. Sesje BGP są zdecydowanie dłuższe i bardziej przewidywalne, niż większość połączeń między dwoma urządzeniami z publicznymi adresami IP.

W Sieci

- *http://elceef.itsec.pl/tcprst.c* – narzędzie użyte do przeprowadzenia przykładowego ataku,
- *http://www.takedown.com/coverage/tsu-post.html* – analiza włamania do komputera Tsutomu Shimomury,
- *http://www.faqs.org/rfcs/rfc1948.html* – RFC-1948, obrona przed atakami na numery sekwencyjne,
- *http://www.faqs.org/rfcs/rfc2827.html* – RFC-2827, obrona przed atakami DoS z wykorzystaniem podszywania pod adresy IP (spoofing),
- *http://www.faqs.org/rfcs/rfc2385.html* – RFC-2385, ochrona sesji BGP przez sygnatury MD5,
- *http://www.ietf.org/rfc/rfc793.txt* – RFC-793, specyfikacja protokołu TCP.

Zakłócenie komunikacji między routerami wykorzystującymi ten protokół będzie miało wpływ na wydajność komunikacji w sieci dla wielu użytkowników.

Metody obrony

Nie możemy poprawić samego protokołu, ale wykorzystać możliwości, które już posiadamy - owszem. Możemy podjąć próby zmniejszenia domyślnego rozmiaru okna TCP, jednak nie jest to zalecane rozwiązanie. Zmniejszymy jedynie ryzyko ataku kosztem szybkości połączeń. Ponadto nie ma gwarancji, że wszystkie używane przez nas aplikacje będą respektowały wprowadzone zmiany.

Filtrowanie

To najbardziej skuteczna metoda obrony. Atak TCP Reset opiera się na możliwości podszycia pod adres jednego z urządzeń. Routery dostawców internetowych (ISP) nie powinny przepuszczać pakietów, których adresy źródłowe nie należą do puli adresów ich sieci. Oczywiście nie ograniczy to możliwości ataku z wewnątrz, ale jednak jest najlepszym możliwym rozwiązaniem. Ponadto ograniczy to ataki innego typu, jak na przykład atak odmowy obsługi (*Denial of Service*) polegający na wysyłaniu wielu pakietów z losowym adresem źródłowym. Wiele przydatnych wskazówek na ten temat zawiera dokument RFC-2827.

Sygnatury MD5 dla połączeń BGP

Protokół BGP posiada obsługę cyfrowego podpisu przy wykorzystaniu algorytmu szyfrowania MD5 (RFC-2385). Włączenie jej spowoduje, że każdy pakiet będzie zawierał dodatkową opcję pod nagłówkiem TCP. Opcja zawiera sumę kontrolną MD5 dla wybranych elementów nagłówka TCP oraz ustalonego wcześniej klucza znanego dla obu stron połączenia. Bez znajomości klucza jest praktycznie niemożliwe zbudowanie pakietu, który przerwie takie połączenie. Większość komercyjnych rozwiązań wspiera obsługę podpisu MD5 w pakietach TCP sesji BGP. W przypadku systemów open source takich jak Linux, większe bezpieczeństwo będzie niestety wymagało większego wysiłku poprzez instalację dodatkowego oprogramowania.

Podsumowanie

Zagrożenie atakiem tego rodzaju naprawdę istnieje. Praktyczna część artykułu miała za zadanie pokazać, że słabość protokołu może wykorzystać każdy, kto dysponuje odpowiednią wiedzą i nadmiarem czasu. Wiele osób jest zdania, że zagrożenie jest niewielkie. Jednak w przyszłości może powstać robak internetowy z powodzeniem wykorzystujący opisaną metodę do zakłócania komunikacji w Internecie.

Robaki mają niewątpliwą możliwość przeprowadzenia rozproszonego działania z wielu miejsc jednocześnie, co uczyni atak jeszcze bardziej skutecznym. Można wiele zrobić, aby zapobiec temu scenariuszowi. Tworzenie wydajnych i jednocześnie bezpiecznych implementacji stosu TCP jest dobrym rozwiązaniem. ●

O autorze

Zajmuje się bezpieczeństwem sieci i systemów komputerowych. Czasami bawi się w programistę tworząc coś, co ma być przydatne, ale różnie z tym bywa. Na co dzień student drugiego roku informatyki.

Kontakt z autorem: *elceef@itsec.pl*