Symulacja w Badaniach i Rozwoju
Vol. 9, No. 1-2/2018

Dariusz PIERZCHAŁA
Wojskowa Akademia Techniczna, Wydział Cybernetyki
ul. Gen. Witolda Urbanowicza 2, 00-908 Warszawa
E-mail: dariusz.pierzchala@wat.edu.pl

Modelowanie symulacyjne wybranych ataków w cyberprzestrzeni

1 Wstęp

Pojęcie „cyberprzestrzeni" funkcjonuje we współczesnym świecie w wielu semantycznych wymiarach. Rozumienie tego słowa jest różne dla inżynierów IT, socjologów, polityków, zwykłych użytkowników Internetu etc. W oficjalnym słowniku pojęć opisujących społeczeństwo informacyjne Komisja Europejska zaproponowała następującą definicję: „Wirtualna przestrzeń, w której krążą elektroniczne dane przetwarzane przez komputery PC z całego świata" [4]. Przyjmiemy w artykule, że cyberprzestrzeń to cyfrowa przestrzeń oparta na sieciach komputerowych, w której użytkownicy gromadzą, przetwarzają i wymieniają informacje. Skonstruowana jest z cyfrowych sieci, będących składowymi globalnej sieci Internet, a istotne jest, że zawierają się w niej systemy informacyjne obsługujące domy, biznes, infrastrukturę krytyczną oraz wspomagające świadczenie szeregu istotnych dla życia usług (finanse, medycyna, …). To sprawia, że jest obiektem coraz częstszych ataków przestępczo-terrorystycznych. Silne uzależnienie współczesnych technik pracy od infrastruktury sprzętowo-programowej cyberprzestrzeni sprawia, że zagadnienia bezpieczeństwa, niezawodności i efektywności sieci komputerowych nabrało szczególnego znaczenia. Nakłada się na to dynamika rozwoju technologii IoT (ang. *Internet of Things*) – wg Mobility Report firmy Ericsson prognozowany wzrost liczby nowych urządzeń działających na zasadzie Internet of Things w Europie w latach 2015–2021 wyniesie 400% (głównie dzięki inteligentnym licznikom energii i samochodom skomunikowanym z Internetem).

O narastającej potrzebie narzędzi programowych, umożliwiających właściwe projektowanie nowych oraz modyfikowanie istniejących sieci, świadczą upubliczniane informacje o skutecznych atakach na zasoby w cyberprzestrzeni, a także o setkach incydentów odnotowywanych przez zespoły CERT. Trendy światowe w dziedzinie badań operacyjnych wskazują, że jednym z najpowszechniej stosowanych podejść w modelowaniu cyberprzestrzeni na poziomie urządzeń, protokołów, usług i użytkowników jest symulacja komputerowa, dająca możliwości ilościowej oraz jakościowej analizy i prognozy. Jest uznaną alternatywą dla eksperymentowania z rzeczywistym systemem lub jego fizycznym prototypem, dla analizy matematycznej, a także zastępuje nieformalne sposoby wnioskowania oparte na doświadczeniu i intuicji. W wyniku rozwoju systemów komputerowych symulacja przeszła ewolucję technologii i algorytmów podobną do typowych systemów IT: od sekwencyjnego eksperymentu symulacyjnego realizowanego na jednej maszynie poprzez równoległy, sieciowy aż po rozproszony w wirtualnych zasobach tzw. chmur obliczeniowych (ang. *cloud computing*). Dla skutecznej symulacji niezbędne są jednakże dedykowane symulatory

komputerowe – adekwatne, ale równocześnie uniwersalne i otwarte na zmiany. Celem prezentowanej pracy jest konstrukcja i zastosowanie takiego symulatora do odwzorowania sieci z urządzeniami aktywnymi i pasywnymi oraz systemami serwerowymi podatnymi na wybrane ataki.

2 Analiza cyberzagrożeń

Umownie nazwana „cyberprzestrzeń" – a w istocie infrastruktura lokalnych sieci włączonych w globalny Internet, z zasobami informacyjnymi i usługami, oparta na pasywnych oraz aktywnych komponentach fizycznych (okablowanie, urządzenia), protokołach i aplikacjach – jest obiektem wyspecjalizowanych ataków mających na celu kradzież informacji, wyłudzenia finansowe czy uniemożliwienie sprawnego działania instytucji. Powszechnie znane są zagrożenia typu: wirus i robak komputerowy, koń trojański, spyware, adware, rootkit, dialer, hoax, axploit, spamming, phishing, sniffing, backdoor czy wreszcie flooding oraz DoS. To właśnie dwa ostatnie według licznych badań są szczególnie uciążliwe z uwagi na łatwość wykonania i relatywnie dotkliwe skutki ich wystąpienia. Podstawowym schematem działania jest tu bowiem przeciążanie (zalewanie) przez wysyłanie ogromnej liczby fałszywych pakietów z próbą nawiązania połączenia. Powoduje to przeciążenie łączy sieciowych oraz serwerów obsługujących żądania. Jeśli atak prowadzony jest jednocześnie z wielu źródeł (komputerów, urządzeń IoT), mamy do czynienia z DDoS (ang. *Distributed Denial of Service*). Zaatakowany skutecznie system komputerowy przydziela zasoby pamięciowe, procesor oraz pasmo sieciowe aż do wyczerpania dostępnych zasobów, co w efekcie prowadzi do przerwy w działaniu lub zawieszenia usług systemu. Stąd podstawowymi strategiami ataków są:

- atak na przepustowość: polega na wykorzystaniu zasobów sieciowych serwera, aż serwer będzie niedostępny na zewnątrz;
- atak na zasoby: polega na wykorzystaniu zasobów systemu komputerowego aż do zablokowania odpowiedzi na poprawne zapytania;
- atak na błędy w oprogramowaniu („exploit"): polega na wykorzystaniu błędu w oprogramowaniu aż do zablokowania lub przejęcia kontroli nad systemem.

Tab. 1. Rodzaje ataków DDoS [6]

Tab. 1. DDoS attack's types [6]

Nazwa ataku	*Typ ataku*	*Zasady ataku*
ICMP Echo Request Flood	Zasoby	Ping Flood - masowa wysyłka pakietów (ping) - powoduje odpowiedź atakowanego serwera (pong) o zawartości, jak pakiet początkowy
IP/TCP Packet Fragment Attack	Zasoby	Wysyłka pakietów IP/TCP odnoszących się do innych pakietów, które nigdy nie zostaną wysłane - przeciążają pamięć atakowanego serwera
IGMP Flood	Zasoby	Masowa wysyłka pakietów IGMP (*multicast*)
TCP SYN Flood	Zasoby	Masowa wysyłka zleceń połączenia TCP
TCP Spoofed SYN Flood	Zasoby	Masowa wysyłka zleceń połączenia TCP poprzez przywłaszczenie adresu źródłowego

TCP ACK Flood	Zasoby	Masowa wysyłka potwierdzeń dla segmentów TCP
SMURF	Przepustowość	Atak ICMP przez broadcast - przywłaszcza adres do przekierowywania różnych odpowiedzi na atakowany serwer
Ping of Death	Exploit	Wysyłka pakietów ICMP wykorzystująca błąd w niektórych systemach operacyjnych
TCP SYN ACK Reflection Flood	Przepustowość	Masowa wysyłka żądań połączeń TCP na dużą liczbę maszyn wraz z przywłaszczeniem adresu poprzez atakowany serwer - w odpowiedzi na zapytania dochodzi do wysycenia przepustowości serwera
UDP Flood	Przepustowość	Masowa wysyłka pakietów UDP bez ustanowienia wcześniejszego połączenia
UDP Fragment Flood	Zasoby	Wysyłka datagramów UDP odnoszących się do innych datagramów, które nigdy nie zostaną wysłane - zapełnia pamięć atakowanego serwera
Distributed DNS Amplification Attack	Przepustowość	Masowa wysyłka zapytań DNS poprzez przywłaszczenie adresu atakowanego na dużą ilość serwerów DNS - amplifikacja odpowiedzi
DNS Flood	Zasoby	Atak z serwera DNS poprzez masową wysyłkę zapytań
HTTP(S) GET/POST Flood	Zasoby	Atak z serwera www poprzez masową wysyłkę zapytań
DDoS DNS	Zasoby	Atak z serwera DNS polegający na masowej wysyłce zapytań z dużej liczby systemów komputerowych

Wraz z rozwojem Internetu Rzeczy zwiększyła się skala masowego atakowania urządzeń podłączonych do IoT (od mobilnych, przez urządzenia SmartTV, aż po typowe urządzenia infrastruktury sieciowej). Wysoka podatność tych dość słabo zabezpieczonych urządzeń czyni z nich łatwy łup dla cyberprzestępców. Mówią o tym różne analizy, np. firmy Trend Micro. Zaatakowane urządzenia IoT stają się następnie zdalnymi źródłami kolejnych ataków typu DDoS. Stopień amplifikacji ataku DDoS (czyli inaczej wielkość odpowiedzi w bajtach na 1 bajt zapytania) może przez to wynieść od setek do tysięcy. Obecnie niemalże każde urządzenie z procesorem i interfejsem sieciowym jest potencjalnym narzędziem ataku DoS – tanim i powszechnym. Technologie służące do ataków można zbudować, pozyskać za darmo albo zakupić – przykładem jest *Low Orbit Ion Cannon* (LOIC), czyli pakiet programowy do ataków typu DoS (a w przypadku wielu atakujących to DDoS) w celu przeciążenie atakowanego serwera dużą ilością pakietów TCP/UDP lub żądań protokołu HTTP. Spektakularne użycia LOIC to m.in.:

- operacja PayBack grupy hakerów LulzSec przeciwko stronom firm i organizacji, które zostały uznane za działające na szkodę portalu Wikileaks;
- ataki na polskie strony rządowe w styczniu 2012 roku jako protest przeciwko uznaniu międzynarodowej umowy ACTA.

Siła ataków DDoS jest ogromna – w wykonanym w 2016 roku ataku na serwis „KrebsOnSecurity.com" został wygenerowany ruch na poziomie 620 Gb/s.

Przeprowadzony również w 2016 roku atak na serwery DNS firmy Dyn miał siłę 1,2 Tb/s, a jego efektem było czasowe zablokowanie dostępu do serwisów internetowych Twitter i Spotify [5].

Atak DoS/DDoS może dotknąć dosłownie każdego użytkownika Internetu, dlatego warto, a nawet należy podjąć środki zapobiegawcze, które zminimalizują straty po jego wystąpieniu. Jawi się zatem kluczowe pytanie: jakie są skuteczne sposoby ochrony przed DDoS? Skuteczna obrona przed atakami DDoS na poziomie sieciowym jest niestety bardzo trudna. Powszechnie zaleca się prewencyjne monitorowanie sprawności infrastruktury, analizę anomalii w ruchu sieciowym, nadmiarowość infrastruktury z możliwością łatwego przełączenia się oraz wykorzystanie rozwiązań odseparowujących ruch DDoS, w tym filtrów w urządzeniach takich, jak firewall. Trafnie przygotowane reguły dla systemów firewall mogą odfiltrować większość ruchu generowanego przez atak prowadzony za pomocą narzędzi podobnych do pakietu LOIC. Współczesne zapory sieciowe umożliwiają realizację bardzo złożonych algorytmów bezpieczeństwa wraz z integracją z systemami IDS (*Intrusion Detection System*) lub IPS (*Intrusion Prevention System*), czyli systemami wykrywania i blokowania ataków w czasie rzeczywistym. Mechanizmy IPS/IDS analizują ruch w sieci z wykorzystaniem metod heurystycznych (protocol analysis w IBM ISS czy preprocessing w Snort) oraz sygnaturowych (wyszukiwanie w pakietach ciągów danych charakterystycznych dla znanych ataków sieciowych).

Działania prewencyjne – ich rodzaj oraz zakres – powinny być oparte nie tylko na niesformalizowanych heurystykach, ale również na dokładnych metodach analitycznych oraz oszacowaniach symulacyjnych. Podobnie jak w przypadku typowego monitoringu w analizach i modelowaniu bada się m.in.:

- wielkość ruchu wchodzącego i wychodzącego;
- parametry serwerów i interfejsów sieciowych;
- obciążenie procesorów i interfejsów sieciowych, ilość operacji na nośnikach danych oraz pamięci operacyjnej.

Badania symulacyjne nabierają szczególnego znaczenia w sytuacji, gdy systemy zabezpieczeń winny być zbadane i przygotowane wszechstronnie – zarówno na ataki o dużej skali (wiele łatwych do zidentyfikowania zapytań w warstwie sieci), jak również na te mniejsze, ale za to bardziej złożone (mniej zapytań w warstwie aplikacji, ale za to istotnie trudniejszych do odróżnienia od typowego ruchu). Służy do tego analiza scenariuszowa, w której badamy modelowany system dla szeregu hipotetycznych sytuacji, w tym także nietypowych warunków. Poprzez eksperymenty symulacyjne wykorzystujące generatory liczb pseudolosowych odkrywa się również wąskie gardła w sieci i infrastrukturze zabezpieczeń, co dla rozważanych ataków DDoS ma największe znaczenie.

Zastosowana w prowadzonych badaniach autorska biblioteka do symulacji dyskretnej zdarzeniowej pozwala na definiowanie i implementację modeli symulacyjnych dla wszystkich klas systemów komputerowych i sieciowych, każdego rodzaju ataku oraz z takim uszczegółowieniem, jakie przyjmie autor modelu.

3 Symulacja dyskretna w pakiecie DESKit

Przyjmiemy, że symulacja komputerowa to ilościowa i jakościowa metoda modelowania w języku formalnym oraz odwzorowania w programie komputerowym strukturalnych i behawioralnych cech systemów (rzeczywistych lub projektowanych), umożliwiająca eksperymentowanie z modelem (zamiast z systemem) i obserwowanie w nim procesów zachodzących w symulacyjnym czasie. Początki symulacji komputerowej to rozwój modeli sieciowych (Carl Adam Petri), metod analizy dynamiki systemów (Jay W. Forrester) a następnie języków i metod symulacji ciągłej i dyskretnej: różniczkowe (Lockheed – Digital Differential Analyzer Simulator DIDAS9), zdarzeniowe (Harry Markowitz, Bernard Hausner – SIMSCRIPT), interakcje procesów (Ole-Johan Dahl i Kristen Nygaard – Algol Simula67, Knuth i McNeley – Algol SOL), selektywnego wykonywania aktywności (John Buxton i John Laski – Fortran CSL) oraz trzy-fazowe (Keith Douglas Tocker – *General Simulation Program* – GSP). Kamieniem milowym był opracowany w 1967 roku w Oslo język Simula67, który poprzez swoją koncepcję klasy i obiektu odszedł od powszechnie stosowanego wówczas podejścia strukturalnego, stając się niezbywalnie protoplastą obiektowości. Odpowiedzią na wciąż rosnące zapotrzebowanie symulacyjne są pakiety rozszerzające języki ogólnego przeznaczenia (np. C#, Java) o usługi symulacyjne (m.in. dynamiczne wyliczanie wartości zmiennych stanu, sterowanie upływem czasu, generowanie liczb losowych, rozpraszanie procesów w sieci). Dzięki takim praktykom każda ewolucja dotykająca języka implikuje również nowe możliwości w aplikacjach symulacyjnych – eksperymenty symulacyjne prowadzone są już nie tylko na pojedynczych stacjach komputerowych, nie tyko w sieciach lokalnych i gridach, ale również według modelu przetwarzania danych opartym na usługach dostarczanych z chmur obliczeniowych.

Podstawowe i najbardziej uniwersalne zastosowanie ma wciąż symulacja konstruktywna (ang. *constructive simulation*) – oparta na formalnych modelach zagregowanych obiektów odwzorowanych w programach komputerowych obsługiwanych przez ludzi (decydentów, analityków, konsumentów wyników). Popularność metod symulacyjnych jako narzędzia w modelowaniu i ocenie wydajności komputerów oraz sieci telekomunikacyjnych znacznie wzrosła w ostatnich latach [2,3,8]. Tabela 2 przedstawia zestawienie wybranych symulatorów sieci komputerowych pod kątem ich przeznaczenia oraz odwzorowanej rzeczywistości.

Tab. 2 Lista pakietów programowych do symulacji sieci
Tab. 2 List of software packages for network simulation

Nazwa	*Zakres symulacji*	*Modelowane elementy*
OPNET	Kolejkowanie usług- LIFO, FIFO; kolejki priorytetowe, *round-robin*	IP, TCP, Ethernet, 802.11, FDDI, ATM oraz wsparcie sieci bezprzewodowych
NetSim	Względne pozycje stacji w sieci, realistyczne modelowanie propagacji sygnału, obsługa kolizji i proces wykrywania	TCP/IP, Ethernet, WLAN, ATM
QualNet	Ocena różnych protokołów	Sieci bezprzewodowe i przewodowe; Sieci WAN

OMNeT++	Opóźnienia, jitter, utraty pakietów	Sieci bezprzewodowe
Ns-2	Przeciążenia, kolejkowanie i algorytmy routingu, multicast	TCP/IP, Protokoły TCP w sieciach przewodowych i bezprzewodowych
GloMoSim	Ocena różnych bezprzewodowych protokołów sieciowych, protokoły MAC	Sieci bezprzewodowe
GTNetS	Śledzenie pakietów, metody kolejkowe, metody statystyczne, generatory liczb losowych	Point-To-Point, Ethernet
Shunra VE	Opóźnienia, jitter, utrata pakietów, ograniczenia przepustowości	TCP;Sieci WAN, Point-to-Point,

Na uwagę zasługują przede wszystkim: komercyjny OPNET (*Optimized Network Engineering Tool*) oraz opensource'owy OMNeT++. Obydwa symulatory realizują model dyskretny (w sensie czasu), a także posiadają szereg modułów, z których zestawia się konstruowaną sieć. Jednakże modułowa architektura, konfigurowalność, bogaty zestaw modułów urządzeń i protokołów „kosztują" – pakiety wymagają maszyn o dużych zasobach obliczeniowych i pamięciowych. Przeciwwagą dla „ciężkich" pakietów są lekkie rozwiązania budowane od podstaw i krojone na potrzeby konkretnego użytkownika. W tym celu niezbędna jest programowa biblioteka do modelowania podstawowych komponentów oraz mechanizmy zarządzania czasem i zdarzeniami symulacyjnymi. Autorska propozycją, wychodzącą naprzeciw takim potrzebom, są dwa pakiety do symulacji dyskretnej czynnościowej (procesowej) i zdarzeniowej: DESKit i DisSim.

W konstrukcji modelu symulacyjnego należy zazwyczaj odwzorować składową statyczną i dynamiczną systemu podlegającego symulacji. Struktura statyczna opisuje obiekty wchodzące w skład modelowanego systemu wraz z ich cechami oraz relacjami łączącymi je w związki. Jest ona wystarczająca tylko w przypadku symulacji statycznej, pomijającej zjawisko upływu czasu. W symulacji dynamicznej niezbędne jest zdefiniowanie modeli formalnych i algorytmów programowych wyznaczania kolejnych stanów obiektów w modelowanym systemie. Jeżeli do modelowania systemu zastosuje się paradygmat obiektowy, to do każdej istotnej mierzalnej cechy (fizycznej lub abstrakcyjnej) systemu należy przypisać atrybut z wybranej klasy obiektów odwzorowującej fragment modelowanego świata [7]:

$O = \{o =< id, c >\}, c \in C^O, id \in N$ – zbiór symulowanych obiektów klasy c identyfikowanych przez *id* o wartości niepowtarzalnej w zbiorze obiektów;

C^O – niepusty zbiór klas modelowanych obiektów;

A_c – niepusty zbiór atrybutów określonych dla klasy obiektów $c \in C^O$;

V_a^c – zbiór dopuszczalnych wartości atrybutu $a \in A_c$ klasy obiektów $c \in C^O$.

W praktyce zbiory C^O oraz A_c przedstawia się jako zbiory numerów odpowiednio modelowanych klas obiektów i ich atrybutów. W dowolnej chwili t czasu symulacyjnego każda składowa stanu systemu, a tym samym wyróżniona cecha systemu, widziana będzie jako czwórka uporządkowana: <*o, a, v, t*>. Zatem stan

modelowanego systemu $S(t)$ będzie zbiorem utworzonym przez wszystkie atrybuty wszystkich obiektów istniejących w chwili symulacyjnej t. W symulacji dynamicznej wartości $v \in V_a^c$ atrybutów $a \in A_c$ wyznaczane mogą być jako skutek: przekształcenia, funkcji zmiany stanu, zdefiniowanych reguł lub wprost danych wprowadzonych z zewnątrz (np. od użytkownika).

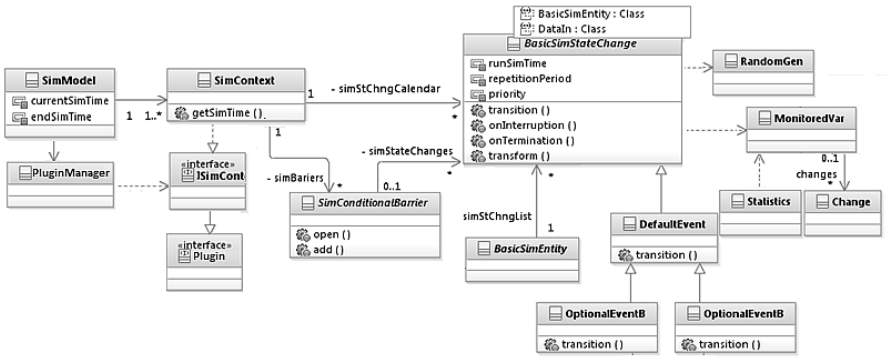
W zastosowaniu jest oczywiście wiele metod odwzorowania dynamiki w modelach symulacyjnych, dając podstawę do ich klasyfikacji z uwagi na sposób odwzorowania upływu czasu symulacyjnego oraz modelowania zmian stanów. Dla nas zasadnicze znaczenie mają jednakże poniższe dwie:

- oparte na zdarzeniach – wartość czasu wynika z wystąpienia kolejnego zdarzenia, czyli zmiany wartości atrybutów stanu, albo zmiany struktury obiektów systemu;
- oparte na czynnościach (procesach).

W *pierwszym podejściu – zorientowanym stricte na zdarzenia* – pod pojęciem zdarzenia e ze skończonego zbioru zdarzeń E rozumiana jest zaplanowana algorytmicznie zmiana stanu obiektu/systemu w określonej chwili czasu symulacyjnego: $e=< t, f_e^S >$, $e \in E$, $t \in T$. Funkcja zmiany stanu f_e^S: T x $S \rightarrow S$ wyznacza stan, w jakim znajdzie się system w chwili t po zajściu zdarzenia e. Mówi się, że zdarzenia opisane różnymi funkcjami zmiany stanu to zdarzenia różnych klas. W symulacji dyskretnej w sensie czasu przyjmuje się następujące uproszczenie modelowe – stan systemu nie ulega zmianie do czasu realizacji kolejnego zdarzenia: $S(t)=S_i$ dla $t \in [t_i, t_{i+1}) \subset T$.

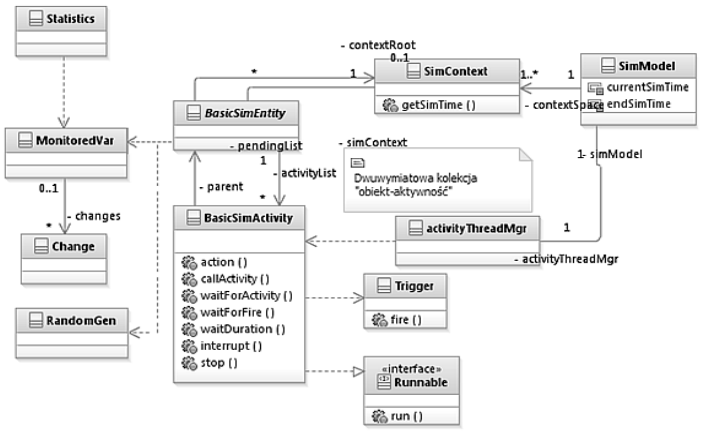
W *podejściu zorientowanym na czynności (procesy)* wprowadzone są pojęcia *czynności i aktywności* obiektów symulacyjnych do modelowania składowej behawioralnej systemu. *Czynność obiektu* jest to niepodzielna akcja trwająca przez pewien niezerowy czas, inicjowana i kończąca się zmianą stanu. Biorąc pod uwagę przyjętą definicję zdarzenia, czynność można definiować za pomocą pary zdarzeń $<e_i, e_j>$, zachodzących w chwilach $t_i, t_j \in T$, $t_i < t_j$. Zdarzenie e_i będzie inicjującym, natomiast e_j kończącym czynność, której czas trwania równy jest $\Delta t_{i,j} = t_j - t_i$. Czas trwania wynika z odległości czasowej między zdarzeniami definiującymi czynność, a biorąc pod uwagę ich zależności przyczynowo-skutkowe, zakończenie czynności może wystąpić bezwarunkowo (trwa nieprzerwanie przez zadany czas) lub warunkowo (przerwanie przed czasem, zależność od warunkowego semafora lub od zakończenia innej czynności). Zmiana stanu systemu następuje w zdarzeniu po zakończeniu jednej czynności i przed rozpoczęciem kolejnej.

Realizacją w języku Java podejścia zdarzeniowego jest pakiet DisSim, zaimplementowany z wykorzystaniem bibliotek odpowiedzialnych za zdarzenia (*BasicSimStateChange*), przesyłanie komunikatów, dynamiczne konfigurowanie obiektów oraz otwartą architekturę wtyczek (*Plugin*).



Rys. 1. Podstawowe klasy pakietu DisSim. Źródło: opracowanie własne
Fig. 1. Basic classes of DisSim package. Source: own preparation

Drugi pakiet – DESKit – jest realizacją w języku Java koncepcji symulacji opartej na czynnościach (*BasicSimActivity*), dla których niskopoziomowym nośnikiem są wątki systemu operacyjnego lub maszyny wirtualnej (*Runnable*), co w praktyce narzuca model programowania współbieżnego.



Rys. 2. Podstawowe klasy pakietu DESKit. Źródło: opracowanie własne
Fig. 2. Basic classes of DESKit package. Source: own preparation

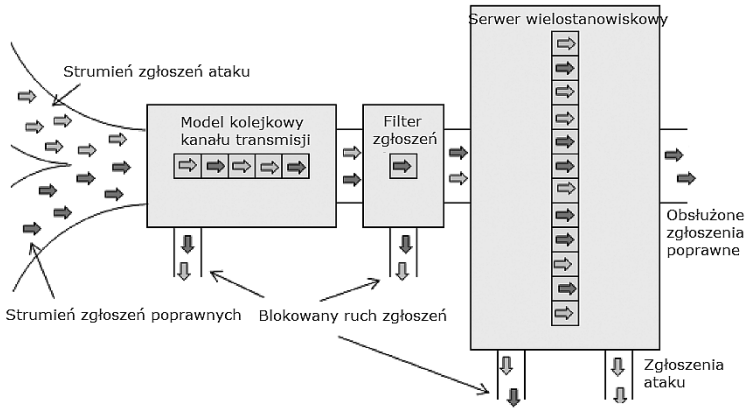
Część klas odpowiadających na przykład za generowanie liczb pseudolosowych (*RandomGen*), monitorowanie zmiennych (*MonitoredVar*), oszacowania statystyczne

(*Statistics*) oraz przesyłanie komunikatów jest bliźniacza w obu wymienionych pakietach.

Podejście oparte na czynnościach (DESKit) uzasadniają ograniczenia pamięci i mocy komputerów i stąd potrzeba zrównoleglania obliczeń w wątkach na wieloprocesorowych architekturach (np. grid, chmura). Natomiast rozwiązanie *stricte* zdarzeniowe (*DisSim*) jest efektywniejsze (w sensie czasu trwania eksperymentu), bardziej niezawodne oraz daje większe możliwości w sterowaniu przebiegiem symulacji.

4 Eksperymenty z modelem symulacyjnym wybranego cyberzagrożenia

Model symulacyjny dla potrzeb eksperymentu opisuje sieć komputerową pewnej instytucji (np. bank, ubezpieczyciel itp.), na serwer której dokonuje się ataku typu DoS. Model matematyczny dla takiego scenariusza z wykorzystaniem teorii sieci masowej obsługi zaproponowano w [1].



Rys. 3. Model ataku DoS. Źródło: wg [1]
Fig. 3. Dos attack model. Source: based on [1]

Przyjmiemy dla potrzeb symulacji następujące założenia dotyczące sieci komputerowej i serwera instytucji. Zgłoszenia poprawne (w [1] zwane uprawnionymi) przesyłane są z otoczenia do kanału transmisji ze strumieniem Poissona o intensywności „lambdaPoprawneZ". Zgłoszenia ataku napływają z otoczenia także zgodnie ze strumieniem Poissona, ale o intensywności „lambdaAtakZ". Czas transmisji zgłoszenia poprawnego w pustym kanale transmisji będzie zmienną losową o rozkładzie wykładniczym z parametrem „lambdaPoprawneT", a czas transmisji zgłoszenia ataku z parametrem „lambdaAtakT". Kanał transmisji obsługuje jednocześnie nie więcej niż K zgłoszeń i odrzuca kolejne zgłoszenia, gdy obsługiwanych jest już K. Po transmisji zgłoszenia są filtrowane przez filtr w firewall, a wynik filtrowania jest losowy – oznaczamy prawdopodobieństwo odfiltrowania (odrzucenia) zgłoszenia poprawnego

przez „pPoprawne”, a dla zgłoszenia ataku przez „pAtak”. Czas przetwarzania pakietu przez filtr firewall jest pomijalnie mały.

Dla zgłoszeń docierających do serwera wielostanowiskowego (M miejsc) rezerwowane jest jedno miejsce na czas jego obsługi. Czasy obsługi zgłoszeń są niezależnymi zmiennymi losowymi o rozkładzie wykładniczym – dla zgłoszeń poprawnych z parametrem „miSPoprawne” oraz dla ataku „miSAtak”. Serwer obsługujący zgłoszenia pracuje według regulaminu RoundRobin podziału czasu procesora z wartością kwantu czasy „dtRR”. Zgłoszenia poprawne odrzucone w kanale transmisji, filtrze w firewall lub serwerze mogą być ponownie zgłoszone przez otoczenie, a prawdopodobieństwo powrotu zgłoszenia oznaczymy przez „pPonownie”. Następne zgłoszenie jest generowane w otoczeniu niezależnie od pozostałych po czasie zadanym losowo z rozkładem wykładniczym z parametrem „miPonownie”.

Implementacja powyższego modelu z wykorzystaniem pakietu DESKit przyjmuje formę obiektową klas i metod – poniżej prezentujemy fragmenty kodu źródłowego:

Czynność zgłaszania

```
while (true) {
  if (czyPoprawne) {
    czasDoKolejnegoZgl = parent.getGenerator().poisson(lambdaPoprawneZ);
    czasObslugi = parent.getGenerator().exponential(miSPoprawne);
  } else {
    czasDoKolejnegoZgl = parent.getGenerator().poisson(lambdaAtakZ);
    czasObslugi = parent.getGenerator().exponential(miSAtak); }
  parent.getKanal().wstawDoKanalu(new Zgloszenie(simTime(), czyPoprawne,
    czasObslugi));
  waitDuration( parent.getGenerator().poisson(mi)); }
```

Czynność transmisji

```
waitDuration(czasTransmisji);
kanal.decLiczbaZgl();
if (zgl.isPoprawne()){
   kanal.getParent().MVczasy_oczekiwaniaP.setValue( simTime()-
     zgl.getCzasWygenerowania());
} else {
  kanal.getParent().MVczasy_oczekiwaniaA.setValue(simTime()-
    zgl.getCzasWygenerowania()); }
kanal.getParent().getFiltr().Filtruj(zgl);
```

Filtrowanie

```
if (zgl.isPoprawne()) {
  pstwo = pPoprawne;
} else {
  pstwo = pAtak; }
if (generator.probability(pstwo)) {
  dt = generator.exponential(parent.getMiPonownie());
  SimActivity sAct = new Aktywnosc_rezglaszaj(parent.getKanal(),dt, zgl);
  SimActivity.callActivity(parent.getKanal(), sAct);
  if (zgl.isPoprawne()){parent.MVodrzuconeP.setValue(
    parent.MVodrzuconeP.getValue()+1);
  } else {
      parent.MVodrzuconeA.setValue(
        parent.MVodrzuconeA.getValue()+1); }
```

```
} else {
  parent.getSerwer().wstawDoBufora(zgl); }
```

Obsługa zgłoszenia w serwerze

```
while (bufor.size() > 0) {
  LinkedList<Zgloszenie> listaObsluzonych = new LinkedList<Zgloszenie>();
  for (int i=0;i<bufor.size();i++){
  Zgloszenie zgl = bufor.get(i);
  if (zgl.getCzasObsl() - dtRR > 0.0) {
    zgl.setCzasObsl(zgl.getCzasObsl() - dtRR);
    waitDuration(dtRR);
    } else {
        waitDuration(zgl.getCzasObsl());
        zgl.setCzasObsl(0.0);
        listaObsluzonych.add(zgl); } }
  Iterator<Zgloszenie> iter = listaObsluzonych.iterator();
  while (iter.hasNext()) {
    Zgloszenie z=iter.next();
    if (z.isPoprawne()) {
      parent.MVczasy_obslugiP.setValue(simTime() -
        z.getCzasWygenerowania());
      parent.liczbaZglObslP++;
    } else {
      parent.MVczasy_obslugiA.setValue(simTime() -
        z.getCzasWygenerowania());
      parent.liczbaZglObslA++; }
    bufor.remove(z);
  }
  listaObsluzonych.clear();
}
```

Dla zadanych wartości parametrów modelu:

K=20; M=50; q=0.4; dtRR=0.05; miPonownie=0.5 [s];

lambdaPoprawneZ=0.4 [s]; lambdaAtakZ=0.6 [s];

miPoprawneT=0.3 [s]; miAtakT=0.1 [s];

pPoprawne=0.02 [s]; pAtak=0.8 [s];

miSPoprawne=0.2 [s]; miSAtak=2 [s]

oraz czasu symulacji równego 600 [s] wybrane oszacowania w jednym z eksperymentów przedstawiają się następująco (tab. 3).

Tab. 3 Wyniki symulacji

Tab. 3 Simulation results

	Wariant 1 – typowa praca, bez ataku	*Wariant 2 – praca w czasie ataku DoS*
Wartość średnia czasu oczekiwania zgłoszenia poprawnego na obsługę [s]	0.8	10.6
Liczba obsłużonych zgłoszeń poprawnych w czasie symulacji	1405	400
Liczba obsłużonych zgłoszeń ataku w czasie symulacji	n.d.	225
Liczba odrzuconych zgłoszeń poprawnych	36	832
Liczba odrzuconych zgłoszeń ataku	n.d.	6070

Wykonany eksperyment pokazuje, iż wykonanie ataku, nawet tak krótkiego (10 minut), prowadzi do istotnych ograniczeń w pracy serwera, co w tym przypadku skutkowało ponad 3-krotnym zmniejszeniem liczby obsłużonych zapytań poprawnych oraz wydłużeniem czasu oczekiwania na obsługę średnio ponad 10-krotnie. Dobierając wartości parametrów symulacji, można odwzorować szereg różnych warunków pracy i scenariuszy ataków oraz zaproponować konfigurację serwera zapewniającą ciągłość jego pracy.

5 Podsumowanie

W ramach prac powstały kolejne wersje autorskich pakietów do symulacji dyskretnej zdarzeniowej i czynnościowej. Skonstruowany został przykładowy model symulacyjny i oprogramowanie, umożliwiające symulację ataku DoS w pewnej instytucji na serwer zabezpieczony przez firewall.

Należy podkreślić, że posiadanie własnego narzędzia do symulacyjnego modelowania cyberprzestrzeni umożliwia zastosowanie szerokiego wachlarza technik i metod badania, prognozowania zdarzeń, a także proponowania zmian poprawiających. Potencjalnymi użytkownikami takich pakietów mogą być zespoły reagowania na zdarzenia naruszające bezpieczeństwo w sieci Internet, np. w strukturze CERT lub SKW.

Literatura

1. Ramanauskaitė S.,Čenys A.: *Composite DoS attack model*, ISSN 2029-2341, s.20-26, 2012
2. Pawlikowski K., Jeong H.-D. J., Lee J.-S. R.: On credibility of simulation studies of telecommunication networks, *IEEE Communications Magazine* nr 40, s. 132-139, 2002
3. Fritz J.: *Network-modelling tools: techworld.com/networking/features/index.cfm*
4. Artykuł: *http://ec.europa.eu/information_society/tl/help/glossary/index_en.htm#c*
5. Artykuł: *http://biznes.t-mobile.pl/pl/artykuly/26-bezpieczenstwo-it/767-6-najwazniejszych-cyberzagrozen-i-sposobow-obrony-przed-nimi*
6. Artykuł: *https://www.ovh.pl/anti-ddos/zasada-anty-ddos.xml*
7. Pierzchała D.: *Symulacja komputerowa – od procedury do chmury*, w monografii ISBN/ISSN: 978-83-7938-038-1, Warszawa, 2014, s. 104-118
8. Dyk M., Najgebauer A., Pierzchała D.: Agent-based M&S of smart sensors for knowledge acquisition inside the Internet of Things and sensor networks, *Int. Inf. and Database Syst.*, LNCS, 9012, Subseries: LNAI, XXXVI, s. 212-223, 2015

Streszczenie

W pracy poruszono zagadnienia ataków w cyberprzestrzeni opartej na sieciach komputerowych (w tym Internet) oraz urządzeniach IoT. Jedną z metod analizy zagrożeń jest modelowanie i symulacja komputerowa, dające możliwości ilościowej i jakościowej analizy oraz prognoz. W ramach prac powstały kolejne wersje autorskich pakietów do symulacji dyskretnej zdarzeniowej i czynnościowej (DiSSim, DESKit). Skonstruowany został przykładowy model symulacyjny i oprogramowanie, umożliwiające symulację ataku DoS w pewnej instytucji na serwer zabezpieczony przez firewall.

Słowa kluczowe: cyberzagrożenia, atak DOS, symulacja dyskretna zdarzeniowa

Simulation modeling of selected attacks in cyberspace

Summary

The presented paper addresses the issues of attacks in cyberspace based on computer networks (including the Internet) and IoT devices. One of the methods for the analysis of threats is computer modelling and simulation – it gives the possibility of quantitative and qualitative analysis and forecasting. As a part of the work, next versions of the original discrete event and functional simulation packages were developed (DiSSim, DESKit). The simulation model and relevant software was implemented to simulate DoS attack on a server protected by firewall in an institution. Our further goal is primarily to create a library with rich content of models of devices and behaviour algorithms.

Keywords: cyberthreats, DOS attack, discrete event simulation