Chapter 6

Discrete-Event Simulation of Botnet Protection Mechanisms

Igor Kotenko, Alexey Konovalov and Andrey Shorov

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50101

1. Introduction

The common use of computers, connected to the Internet, as well as insufficient level of security, allow malefactors to execute large-scale infrastructure attacks, engaging in criminal activity a huge number of computing nodes. Attacks of such type have been traditionally performing by botnets. There are examples of successful large-scale attacks fulfilled by armies of bots. For example, attacks such as distributed denial of service (DDoS), aimed at government websites of Estonia in 2007 and Georgia in 2008 had led to the practical inaccessibility of these sites for several days. In 2009 and 2010 spying botnets "GhostNet" and "Shadow Network" have been occurred in many countries around the world.

Further research of these botnets has shown their presence on governmental servers, which contain important sensitive information. In 2009 a malware "Stuxnet" was discovered, which was capable to affect SCADA-systems and steal intellectual property of corporations. Report "Worldwide Infrastructure Security Report", published by Arbor Networks in 2010, shows that the total capacity of DDoS attacks in 2010 has grown considerably and has overcome the barrier of 100 GB/sec. It is noted that the power of DDoS-attacks has grown more than twice in comparison with 2009 and more than 10 times in comparison with 2005.

On this basis, it becomes obvious that existing modern botnets are a very important phenomenon in the network security. Thus, the task of researching botnets and methods of protection against them is important. One of the promising approaches to research botnets and protection mechanisms is simulation.

This paper is devoted to investigating botnets, which realize their expansion by network worm propagation mechanisms and perform attacks like "distributed denial of service" (DDoS). The protection mechanisms against botnets are of top-priority here. The main results of this work are the development of integrated simulation environment, including

the libraries for implementing models of botnets and models of protection mechanisms. As distinct from other authors' papers (for example, [12, 13]), this paper specifies the architecture of the integrated simulation environment and describes the set of conducted experiments on simulation of botnets and protection mechanisms against them.

Comparing with previous research, the architecture of the integrated simulation environment has been substantially revised - more emphasis has been placed to extend the libraries of attacks and protection mechanisms. Also in this version of the simulation environment, we have applied a hierarchical component-based way of representation of architecture. The main attention in the paper is paid to the set of experiments, which provided the opportunity to compare protection methods against botnets on different stages of their life cycle.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents the architecture of the integrated simulation environment developed. Section 4 considers the attack and defense models. Section 5 contains the description of implementation and main parameters and the plan of the experiments. Section 6 describes the results of experiments. Concluding remarks and directions for further research are given in Section 7.

2. Related work

The current work is based on results of three directions of research: analysis of botnets as a phenomenon occurred in the Internet [2, 6, 8, 19, 21], including the studies of botnet taxonometry, approaches of creation and improving the techniques for counteraction against modern botnets, and enhancement of concepts and methods for efficient modeling and simulation of botnet infrastructure and counteraction.

At present moment using public proceedings we can find many interpretations of different aspects of botnet functionality. A group of researches, related to analysis of botnet as a network phenomenon, defines botnet lifecycle [6, 21], which is consisting of several stages: initial infection and spreading stage, stage of 'stealth' operation and attack stage. Centralized [21] and decentralized [6, 8, 34] kinds of architectures are considered as results of investigation of feasible node roles, and different types of botnet arracks are described.

The investigations, devoted to botnet counteraction methods, may be conditionally divided into two logical groups: methods, which are based on identification of predefined signatures [28], and methods which rely on detection of local and network anomalies [3, 10, 18, 32]. The second group of methods has a significant advantage against first group in ability to detect unknown threats not having specific knowledge of their implementation [15]. On the other hand, the second group is much more resource consuming and more subjected to false positive and false negative errors.

Due to significant differences of botnet lifecycle stages, the combined protection methods are used extensively which take into account specificities of each stage.

Defense techniques "Virus Throttling" [37] and "Failed Connection" [4] are used to oppose botnet propagation on spreading stage. Such techniques as Threshold Random Walk [20] and Credit-based Rate Limiting also require consideration.

Beyond many types of botnets attacks, we studied botnets which implement DDoS as an actual attack stage. We considered protection methods for different phases of DDoS attacks. Approaches Ingress/Egress Filtering and SAVE (Source Address Validity Enforcement Protocol) [17] are used as attack prevention mechanisms. They realize filtering of traffic streams for which IP spoofing was detected. Moreover, such techniques as SIM (Source IP Address Monitoring) [23] and Detecting SYN flooding [35] were taken into consideration as methods for discovering DDoS attacks.

We also investigated protection methods destined to detect botnets of different architectures. Botnet architecture is defined by the applied communication protocol. At present moment IRC-, HTTP- and P2P-related botnet architectures [21] are important for consideration.

Research on botnet modeling and simulation is based on a variety of methods and approaches. A large set of publications is devoted to botnet analytical modeling. For instance, a stochastic model of decentralized botnet propagation is presented in [26]. This model represents a botnet as a graph. Nodes of this graph represent the botnet states, and edges depict possible transitions between states. D.Dagon et al. [5] proposes an analytical model of global botnet, which describes dependencies between the activities of botnet nodes and the time zone for location of these nodes.

Another group of studies uses simulation as a main tool to investigate botnets and computer networks in general. Studies in this group mainly rely on methods of discrete-event simulation of processes being executed in network structures [29, 36], as well as on trace-driven models initiated by trace data taken from actual networks [22]. G.Riley et al. [25] use the GTNetS simulation environment to build network worm propagation model. A.Suvatne [30] suggests a model of "Slammer" worm propagation by using "Wormulator" [14] simulation environment. M.Schuchard [27] presents simulation environment which allows to simulate a large-scale botnet containing 250 thousands of nodes. Gamer at al. [7] consider a DDoS simulation tool, called Distack. It is based on OMNeT++ discrete simulation system. Li at al. [17] use own simulation environment and testbeds to estimate efficiency, scalability and cost of implementation of protection mechanism SAVE.

Other techniques, which are very important for investigation of botnets, are emulation, combining analytical, packet-based and emulation-based models of botnets and botnet defense (on macro level), as well as exploring real small-sized networks (to investigate botnets on micro level).

This paper describes the approach, which combines discrete-event simulation, component-based design and packet-level simulation of network protocols. Initially this approach was suggested for network attack and defense simulation. In the present paper, as compared

with other works of authors, the various methods of botnet attacks and counteraction against botnets are explored by implementing comprehensive libraries of attack and defense components.

3. Simulation environment architecture

The proposed simulation environment realizes a set of simulation models, called BOTNET, which implement processes of botnet operation and protection mechanisms.

With narrowing the context of consideration, these models could be represented as a sequence of internal abstraction layers: (1) discrete event simulation on network structures, (2) computational network with packet switching, (3) meshes of network services, (4) attack and defense networks.

Specification of every subsequent layer is an extended specification of the previous one. Enhancement of specification is achieved by defining new entities to the preceding layer of abstraction. Proposed view on semantic decomposition of BOTNET models is shown in Fig.1. Hierarchy of abstraction layers reproduces the structure of simulation components and modules.

Simulation environment relies on several libraries - implemented by authors of the paper and third party libraries. Functionality of each library matches to the appropriate layer of abstraction. The library, which is related to attack and defense networks layer, is implemented by the authors. All components of simulation environment are implemented in C++ programming language with standard runtime libraries.

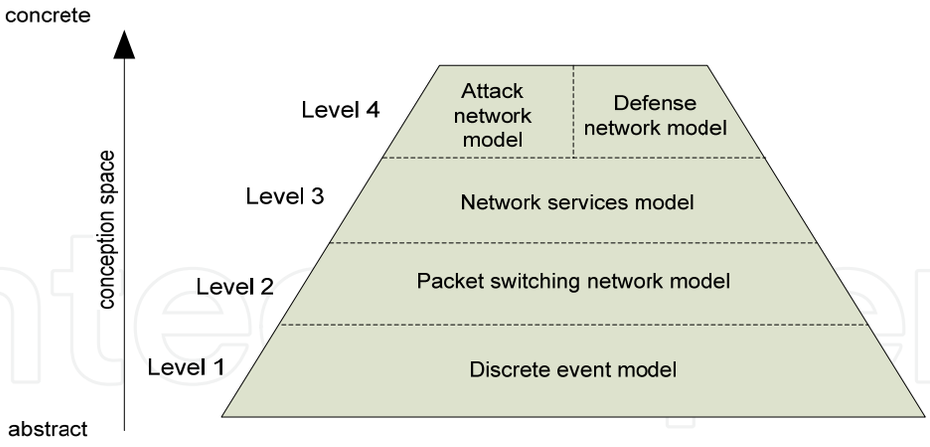


Figure 1. Hierarchy of Abstraction Layers

The diagram representing the relations between layers of abstraction and implementation layers (libraries) is shown in Fig.2. Each particular library provides a set of modules and components, which are implementations of entities of appropriate semantic layer. Any given library can rely on the components exported by the libraries of the previous layer and can be used as a provider of components needed for the subsequent layer implementation.

The first layer of abstraction is implemented by use of discrete event simulation environment OMNET++ [31]. OMNET++ provides the tools for simulation of network structures of different kinds and processes of message propagation in these structures.

The library INET Framework [11] is used for simulation of packet-switching networks. This library provides components implemented as OMNET++ modules and contains large variety of models of network devices and network protocols for wired and wireless networks.

Simulation of realistic computer networks is carried out by using the library ReaSE [24]. The library is an extension of INET Framework [11]. It provides tools for creating realistic network topologies which parameters are statistically identical to parameters of real computer networks topologies. ReaSE includes also a realistic model of network traffic, modeled at the packet level [16, 38]. Models of network traffic are based on the approach, presented in [33]. This approach allows generating packet level traffic with parameters, which are statistically equivalent to the traffic observed in real computer networks.

Simulation of target domain entities is committed through the set of components implemented by the authors. These components are integrated into the BOTNET Foundation Classes library (Fig.2). This library includes models of network applications belonging to botnets of various types and appropriate defense methods.

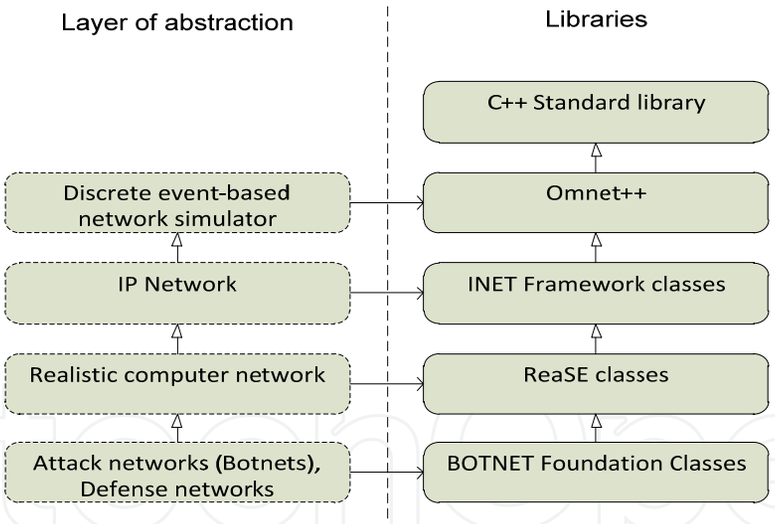


Figure 2. Relation between Abstraction and Implementation Layers

On the fourth layer of abstraction, all set of components of the target domain is divided into two groups related to attack and defense network correspondingly.

The first group contains the components (1) responsible for propagation of attack network, (2) supporting attack network on the stage of 'stealth' operation, (3) hampering detection and suppression of attack network, and (4) executing DDoS attacks.

The group of defense network includes the components (1) detecting and suppressing of attack network during every stage of its lifecycle, (2) carrying out management and control

of defense network and (3) providing robustness of defense network (these components realize protocols of centralized and decentralized overlay networks).

According to the structure of the scenarios, the behavior of BOTNET models is defined by the set of conditionally independent network processes. The model of legitimate traffic is based on approach described in [33], and implemented by the components of ReaSE library [24].

4. Attack and defense models

Attack network model specifies a set of activities generated by attack network. In the current work we implemented this model by three relatively independent sub-models: the propagation model, the management and control model, and the attack phase model.

The *model of botnet's propagation* implements a scenario of expanding botnet over the computer network. The main goal of this scenario is the support of the ongoing process of involvement of new nodes into the botnet. In this paper, such scenario is presented as a model of the bot-agent code propagation by the means of the network worms of various types.

Participants of the botnet propagation scenario are the "IP Worm" and "Vulnerable App" components, which are the model of a network worm and the model of vulnerable network server, respectively (Fig. 3).

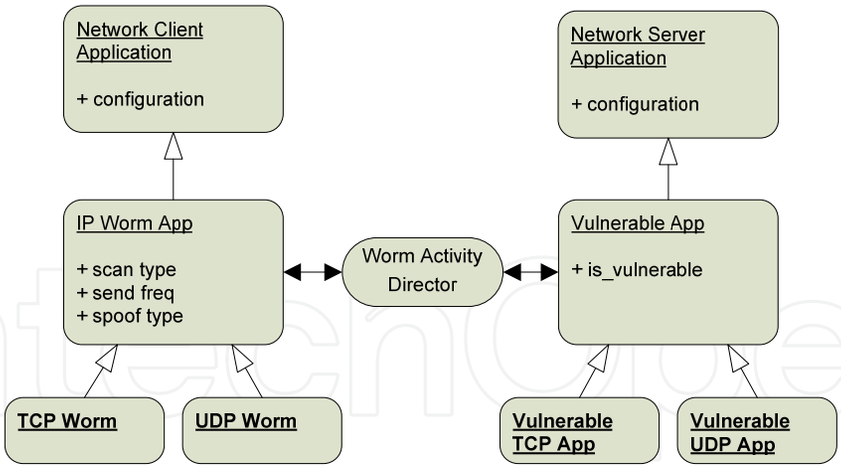


Figure 3. Static diagram of components involved in the botnet propagation scenario

The component "IP Worm" is a model of client application that responsible for sending malformed network packets in order to compromise possible vulnerable hosts. The parameters of this model are: the algorithm of victim's address generation, the method of source address spoofing, and the frequency which is used to send malformed packets. The payload of the malformed packet includes the network address of the server, which is supposed to be the command center of the growing botnet.

Component «Vulnerable App» is a model of the vulnerable server, which is prone to be infected by the receiving malformed packed.

The component interaction diagram is shown in Fig.4.

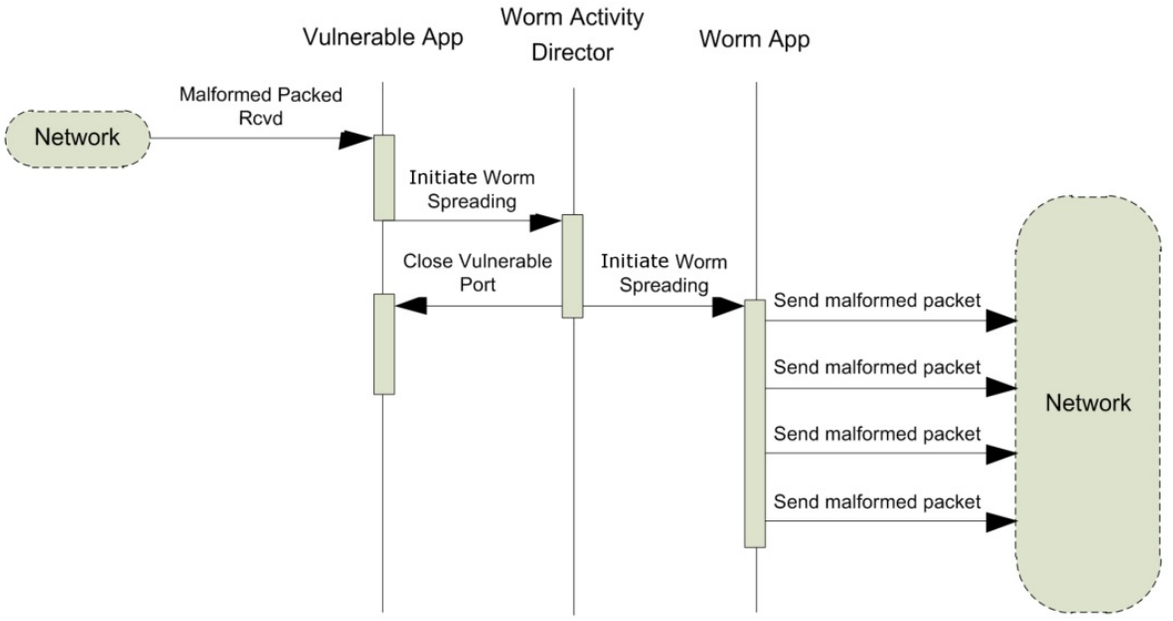


Figure 4. Component interaction diagram for the botnet propagation scenario

All communication between the models of network applications is carried out via instant messaging. The core component that responsible for the messaging support is the "Worm Activity Director". In case of receiving a malformed network packet, the model of vulnerable network service notifies the component "Worm Activity Director" immediately.

Then the component "Worm Activity Director" decapsulates the addresses of the hosts which are supposed to be command centers and sends the message which contains the information about servers to the "Worm App" component. The event of receiving such message by "Worm App" component is considered as a signal of the transition to the infecting state and as the instruction to start spreading a worm from the given host. In the current research we implemented TCP and UDP based worms and vulnerable network applications.

The *model of botnet control* implements the botnet control scenario. The goal of this scenario is to provide the persistent controllability of the whole set of botnet nodes. Such process includes the procedures for support of the connectivity of the nodes and methods that are supposed to make the botnet responsible on the commands being issued by the bot-master.

The model of botnet control represented in this paper implements two types of architectures.

The first type of the botnet is based on the IRC protocol. It is a classic implementation of a centralized botnet with a centralized command centers.

Another type of botnet is based on some implementation of P2P-protocol. Such botnets belong to a group of decentralized botnets [34]. Library component BOTNET Foundation Classes includes components that are common to both types of architectures and components specific to each of them.

Classes include components that are common to both types of architectures and components specific to each of them.

Static diagram of components involved in this scenario shown in Fig. 5.

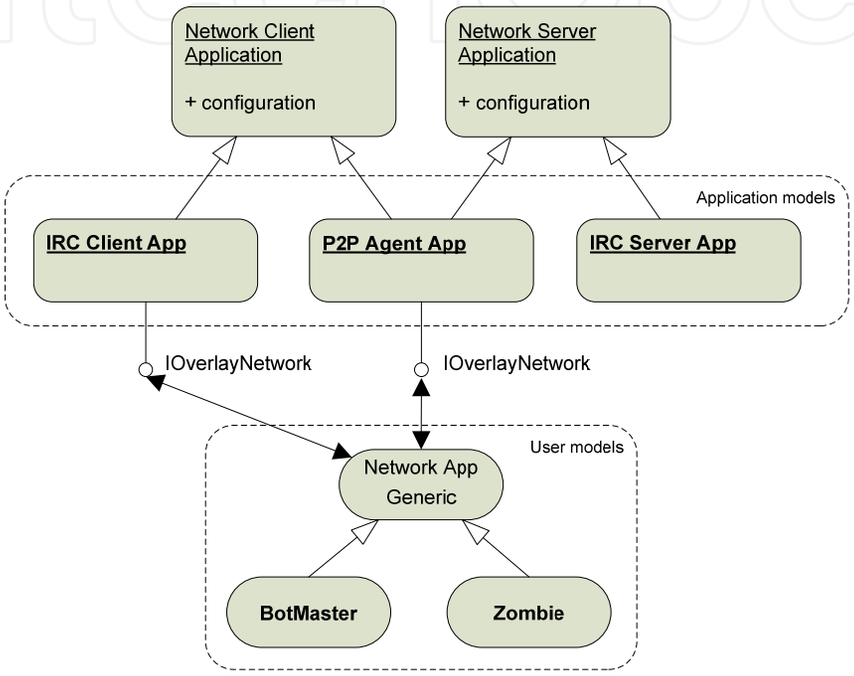


Figure 5. Static diagram of components involved in the botnet control scenario

The main components that are directly responsible for connectivity of the whole botnet, are models of network applications that implement the corresponding application protocols.

In the case of a centralized network, there is a subset of nodes – command centers – which are responsible for management of the rest of botnet nodes. The model of network service that corresponds to command center, is represented as a component "IRC Server App". This component is a particular implementation of a generalized model of the network server. It provides a partial implementation of the server side of the IRC protocol. This implementation is sufficient to reproduce the essential aspects of network behavior. Clients of such server are the "IRC Client App" components that are particular implementations of the generalized model of the network client. These components implement the client side of the IRC protocol as well.

The model of the decentralized botnet control is represented by the "P2P Agent App" component. This component is an implementation of the P2P-protocol client. Regarding to the P2P protocol specificity this component is a particular implementation of both the network server and client applications at the same time.

The essential point is that the components “IRC Client App” and “P2P Client App” are the models of network clients that implement the corresponding application layer protocols. The way to manipulate such clients is performed by the means of the component that represents the model of the user. The commonality that is inherent in the models of the clients allows distinguishing invariant interface for manipulating models of clients by the models of the user. This interface also permits to separate the business logic of the network application from a component that implements the communication protocol itself. Thus, it is possible to apply the components “IRC Client App” and “P2P Client App” over a variety of different protocols without any change of business logic of the clients.

The user models are implementations of a generic user model represented as a component “Network App Generic”. The generic user model is an abstract model, which communicates in the network through the application layer protocol. The user model interacts with the protocol by the “IOverlayNetwork” interface. The specific implementations of the generic user model directly specify the logic of network activity. In the control scenario, two types of specific user models are realized: a bot-master model (“BotMaster”) and a model of bot-agent, which is located on the “zombie”-nodes.

In the present work, in the botnet attack scenario an attack Distributed Denial of Service (DDoS) is simulated. The signal to begin the attack is the special command of the botnet master. The specific ways to implement this attack are “TCP Flood” and “UDP Flood”. These components are directly related to the component representing the network layer protocols in the node model. Fig. 6 shows the diagram of components that implement the “TCP Flood” attack. This figure shows the structure of the node which is included in the centralized botnet.

The “zombie”-node model consist of: (1) the IRC-client model that realize communication between the node and the rest of the network, (2) the zombie-agent model that implements the communication protocol between the zombie node and the bot-master, and (3) the model of the TCP Flood component that realize the attack of TCP Flood type. The component that implements the TCP Flood attack and the IRC-client use the services provided by the module realizing the network layer protocols (TCP, IP, and UDP).

The component interaction diagram for components involved in the botnet attack scenario is shown in Fig. 7.

The bot-master initiates the command to begin the attack and sends it to zombies. This command includes the victim address in data field. Then the message containing the command is delivered to the zombies by using network protocols. The message is transmitted to the “zombie” components for processing in accordance with the logic of the control protocol.

The zombie component identifies the command, retrieves the victim address and notifies the component “IP Flood” to switch to the attack mode. The attach target is the victim node with the given address. The time to finalize the attack is determined by the logic of its implementation.

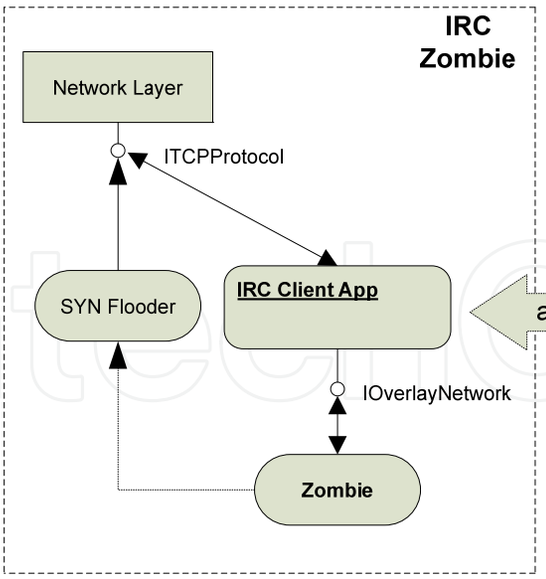


Figure 6. Diagram of components involved in the botnet attack scenario.

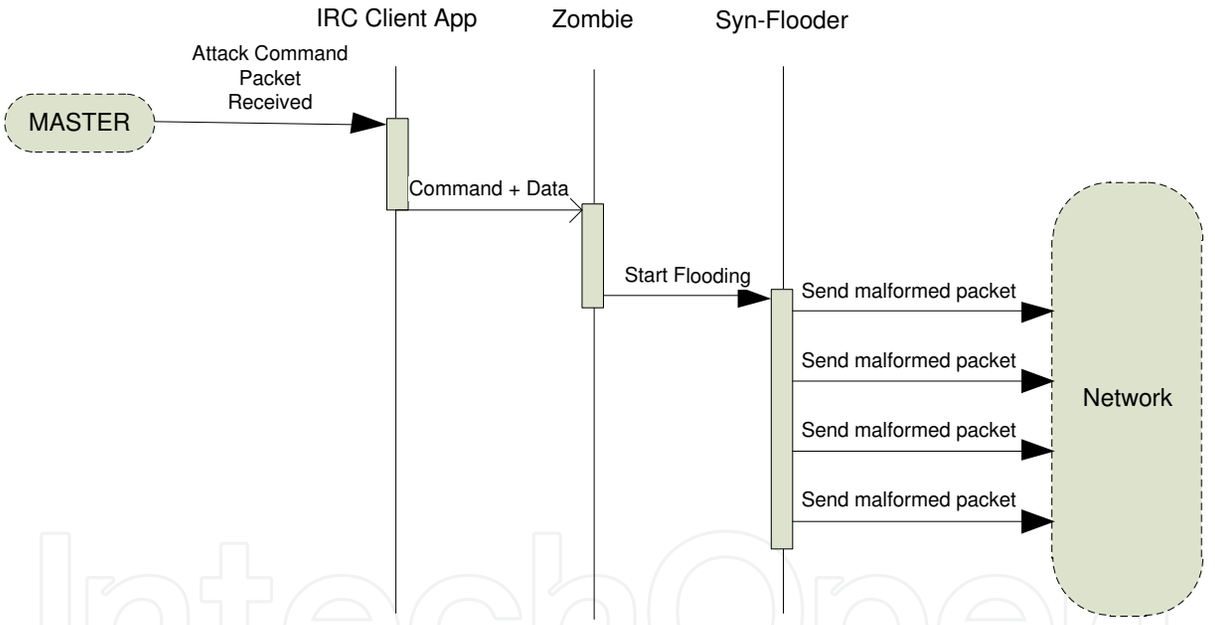


Figure 7. Component interaction diagram for components involved in the botnet attack scenario

For each activity of attack network, the *defense network* is performing the opposite activity, aiming to suppress corresponding activity of attack network. Therefore, the defense network model is implemented by three following sub-models: counteraction of attack network propagation, counteraction of attack network management and control, and counteraction of DDoS attack. In addition to actions against attacking network, protective network also performs some steps destined to ensure its own robustness.

Components that realize the entities of attack and defense networks are given in Table 1.

Module	Description
Attack network modules	
"Botnet Master"	Model of botmaster application
"Bot Client"	Model of zombie client application
"Worm"	Model of network worm application
"Vulnerable Application"	Model of vulnerable network application
"IRC client"	Model of IRC client application
"IRC Server"	Model of IRC server application
"P2P Agent"	Model of P2P client application
"UDP Flooder"	Model of UDP flooding application
"SYN Flooder"	Model of SYN flooding application
Defense network modules	
"Filtering router"	Model of router for filtering network traffic
"Failed Connection filter"	Traffic filter based on "Failed Connection"
"Worm Throttling filter"	Traffic filter based on "Worm Throttling"
"HIPC filter"	Traffic filter based on "Source IP Counting"
"IRC Monitor"	IRC traffic monitor
"IRC Relationship filter"	IRC related traffic filter based on "Relationship" metric [1]
"IRC Synchronization filter"	IRC related traffic filter based on "Synchronization" metric [1]
"Hop-Count Filter"	Traffic filter based on "Hop-Count Filtering"
"SIMP Filter"	Traffic filter based on "Source IP Address Monitoring"
"SAVE Filter"	Traffic filter based on "Source Address Validity Enforcement"

Table 1. Modules of BOTNET Foundation Classes

5. Implementation and parameters of experiments

An example of the simulation environment user interface in one of the experiments is shown in Fig.8. The main panel and control elements are in the upper-left corner of user interface. Main panel shows components, which are included in BOTNET models.

Control elements allow user to interact with these components. The model time control elements are presented optionally on the main panel. These elements allow, for example, to execute model step-by-step or in the fastest mode. There are also control elements, which allow performing efficient search of the appropriate instance and editing its state.

Fragment of the modeled network is also shown in Fig.8 (at the bottom left). Routers models are depicted as cylinders with arrows, and hosts models are represented as computers of different colors. Color represents the node state. Blue color is used for the incoming nodes, which have vulnerabilities.

The legitimate nodes without vulnerabilities are not colored. The view window of one of hosts (at the bottom right) and the editing window of the "router" object parameters (at the upper-right) are also shown in Fig.8.

Network topology and configuration are modeled on the two levels of details.

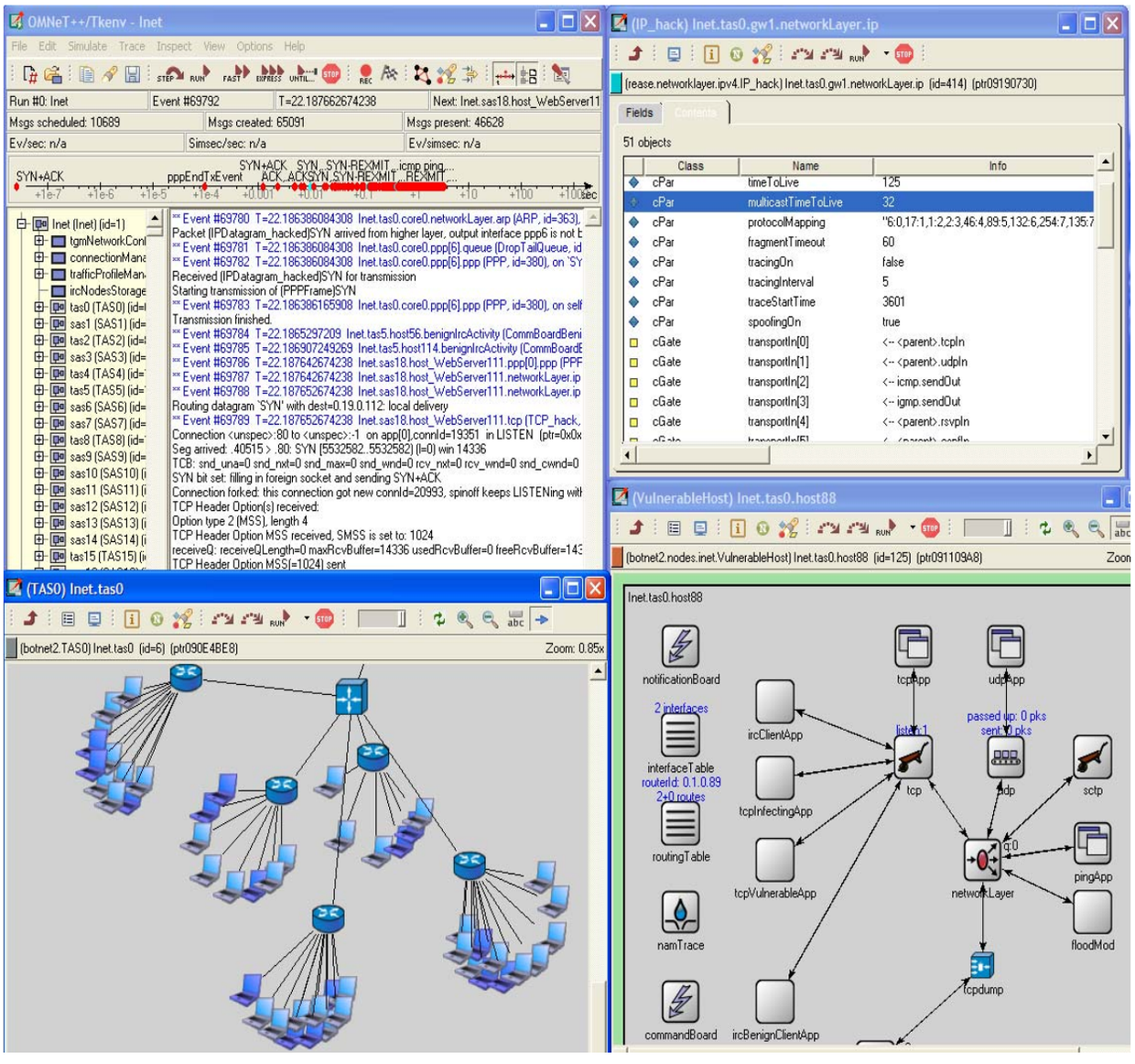


Figure 8. User Interface of Simulation Environment

On the first level the network topology is modeled on the level of the autonomous systems (AS). We used technique of positive-feedback preference (PFP) [38] to model computer network topology on the autonomous system level.

The networks which consist from 30 autonomous systems (AS-level topology) were modeled in experiments (Fig. 9). To generate the graph of the autonomous systems level the following parameters were used: threshold to take AS nodes as transit (Transit Node Threshold= 20); number of new nodes connections (P=0.4); assortative level of the generated network, which characterizes the level of nodes preference depending on their connectivity after addition of the new node to the network (Delta=0.04) [38].

Connection of transit AS is made via communication channel with bandwidth dr=10000 Mbit/s and delay d=50 milliseconds. Connection of limited AS is made with dr=5000 Mbit/s and d=20 milliseconds.

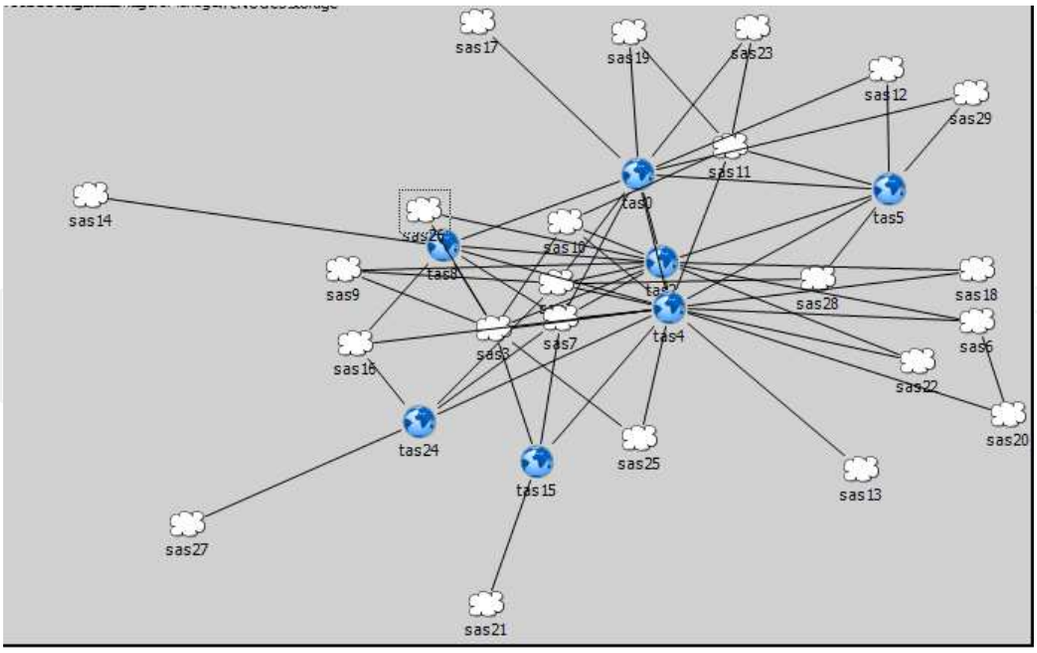


Figure 9. AS-level topology

On the second level the router-level topology is modeled for each AS (Fig. 10).

In this work we use HOT-model (Heuristically Optimal Topology) [16] with the following parameters: number of routers - from 5 to 20; the part of the core routers in the total number of routers - 1%; number of hosts on the router - from 5 to 12; connectivity level of core routers - 0.2.

Core routers are connected via communication channel with bandwidth dr=2500 Mbit/s and delay – 1 milliseconds, communication of gateways with core routers - dr=1000 Mbit/s and delay - 1 milliseconds, connection of gateways with edge routers - dr=155 Mbit/s and delay - 1 milliseconds, connection of edge router with servers - dr=10 Mbit/s and delay - 5 milliseconds. Connection of edge router with client nodes is as follows: to node - dr=0.768 Mbit/s and delay - 5 milliseconds, from node - dr=0.128 Mbit/s and delay - 5 milliseconds.

On the base of the parameters provided above, different networks were generated, including network with 3652 nodes (which is used for experiments described). 10 of these nodes are servers (including one DNS-server, three web-servers and six mail servers). 1119 nodes (near 30% from the total number) have vulnerabilities.

Also the node “master” is defined in the network. It works as the initial source of worm distribution and the initiator of botnet management commands. All nodes in the subnets are connected via edge routers. Root router “gateway” is defined in every subnet. Subnets are united via this router. User models, which send requests to the servers, are installed on the client nodes.

It is the way to create legitimate traffic. Model of the standard protocol stack is installed on each node. This stack includes PPP, LCP, IP, TCP, ICMP, ARP, UDP protocols. Models of the network components (which implement appropriate functionality) can be installed

additionally depending on the nodes functional role. The experiments include investigation of botnet actions and defense activities on the stages of botnet propagation, botnet management and control (reconfiguration and preparation to attacks) and attack execution.

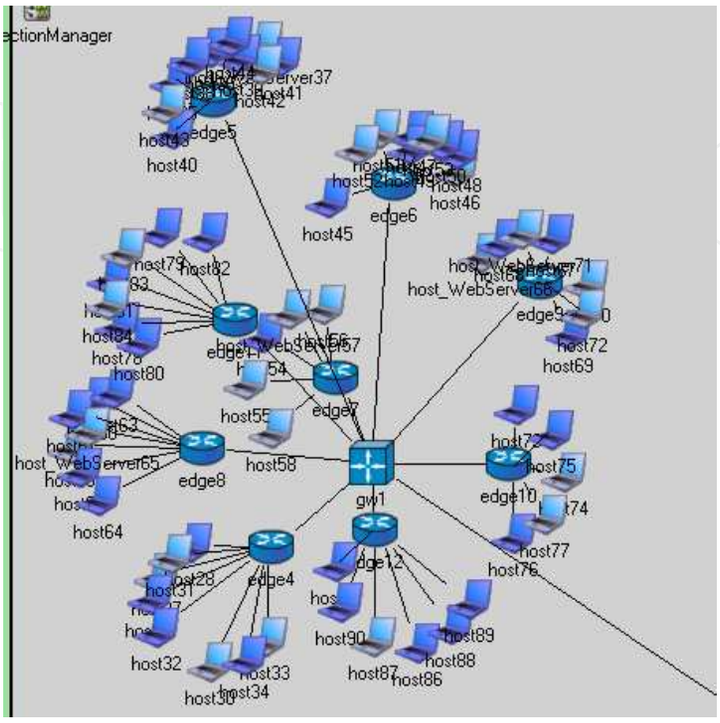


Figure 10. Router-level topology

6. Experiments

As part of our research, a set of experiments was performed. They demonstrate the operability of the developed simulation environment and main characteristics of botnets and defense mechanisms investigated.

a. Botnet propagation and defense against propagation

At the 100th second of the model time, the bot master initiate scanning the network for vulnerable hosts using one of network worm techniques. At the same time it connects to the public IRC server and creates new communication channel, thus turning into kind of "command center". We adjust the frequency of network scanning to 6 packets per second in our experiment. Random scanning on a range of predefined IP addresses is used. In case of some host getting compromised it becomes the "zombie". "Zombie" connects to public IRC server, which is "command center", and reports to bot master of its successful integrating to the botnet infrastructure and readiness to process further orders. Also "Zombie" starts to scan the network for vulnerable hosts the same way as bot master did initially.

To protect against botnet propagation the protection mechanism based on "Virus Throttling" is used. It has the following parameters - the source buffer contains 300 traffic

source addresses; the buffer operates by FIFO principle. For each new source address one slot of the buffer is allocated. The buffer includes up to 5 authorized destination addresses. If the buffer is full, one of its slots can be released every 5 seconds by FIFO method and it is allowed to connect to a new remote host. This protection mechanism is installed on routers.

Several experiments were performed. Fig.11 shows the dependencies of the number of infected hosts from the botnet propagation time for cases without protection and with protection set at 30%, 50% and 100% of the routers.

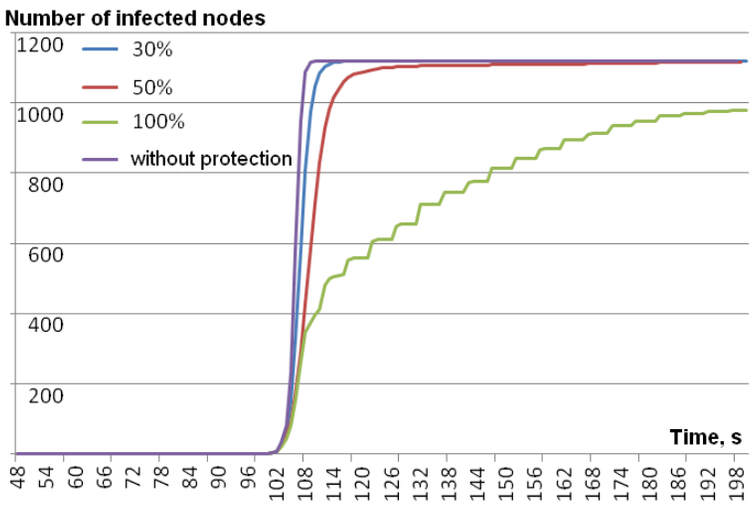


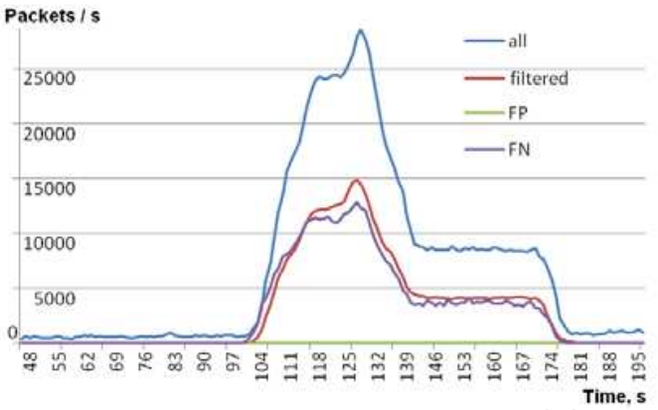
Figure 11. Number of Infected Hosts when using Virus Throttling

We analyzed in the experiments the dependencies of number of false positive rate (FP, when the legitimate packet is recognized as malicious), false negative rate (FN, when the malicious packet is not detected), and truth positive rate (TP, when the malicious packet is detected) from the botnet propagation time.

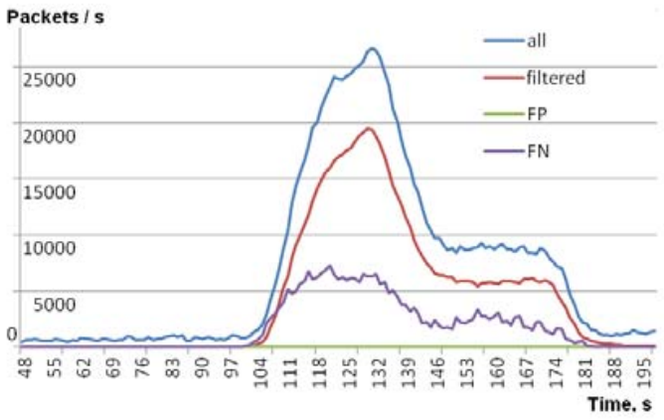
It was shown that the numbers of FP and FN are just slightly different under a small number of established protection mechanisms (30%) and limiting the source buffer up to 300 addresses. This occurs because Virus Throttling passes the packets from infected nodes, which were previously included in the source buffer, but were jammed with new infected node addresses.

When the number of protection mechanisms increased, the number of FN is significantly reduced.

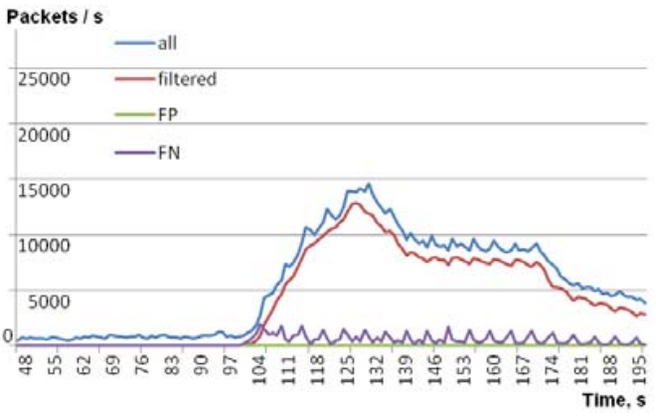
Fig.12 shows the dependencies of the volume of total and filtered traffic, as well as the numbers of FP and FN rates from botnet propagation time using Virus Throttling at 30% (a), 50% (b) and 100% (c) routers, respectively.



(a)



(b)



(c)

Figure 12. Main Characteristics of Virus Throttling

Fig.12 shows that when we increase the number of nodes with Virus Throttling from 30 to 50% the filtered traffic also increases, although the total amount of traffic generated by worms decrease slightly. In the case of full (100%) coverage of Virus Throttling the volume of the traffic generated by network worms and the amount of FN is significantly reduced.

Fig.13 shows the dependencies of the percentage of filtered legitimate traffic related to all legitimate traffic from the botnet propagation time, when Virus Throttling are used at 30%, 50% and 100% of the routers.

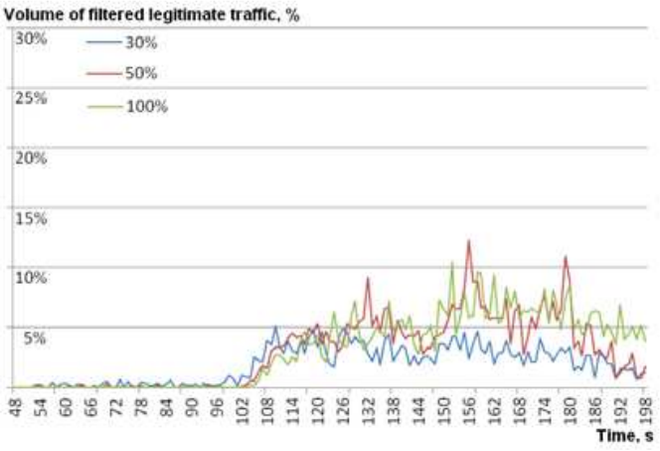


Figure 13. Volume of Filtered Legitimate Traffic when using Virus Throttling

To investigate several other protection mechanisms, we performed the same set of experiments as for Virus Throttling. For example, for "Failed Connection" technique, relative high levels of TP in all dependencies indicate that this technique allows filtering a large number of malicious packets.

However, Failed Connection does not allow significantly constraining the botnet propagation under current parameters of experiments. Such characteristics, as the relation of the number of vulnerable hosts to legitimate ones, the method of vulnerable hosts scanning and threshold for decision making, have a great impact on the quality of this technique.

b. Botnet Management and Protection against Botnet on this Stage

On this stage of botnet life cycle we investigated mainly the protection technique proposed by M. Akiyama et al. [1]. This technique involves monitoring of IRC-traffic, passing through the observer node, and subsequent calculation of the metrics "Relationship", "Response" and "Synchronization", based on the content of network packets. Metric "Relationship" characterizes the distribution of clients in IRC-channel. Too high value of this metric is considered as abnormal.

For example, the threshold for this metric can have a value of 30, 100 or 200 clients per channel. If the threshold is exceeded, the packets related to this IRC channel are filtered. The metric "Response" is calculated as the distribution of response time to the broadcasting request. The metric "Synchronization" characterizes the synchronism in group behavior of IRC clients. Consider the examples of different experiments.

IRC traffic monitoring and relationship metric calculation. IRC traffic is monitored by using "observer" components, which are installed on the core routers of network segments. Information about IRC channel and its clients is defined by analysis of IRC packets. Then, based on data obtained, the relationship metrics of observed channels are calculated in real

time. It is assumed that the data, obtained from observer components, will strongly depend on the location of the observer in relation to main IRC flows, merging near the network segment that contains the IRC server.

Table 2 shows a part of observed relationship metrics for IRC channels in various network locations. There are data for the botnet control channel (Irc-bot) and two channels for legitimate IRC communication (Irc-1 and Irc-2). The number of clients in the Irc-channel 1 is 10; the number of clients in Irc-channel 2 is 9. For legitimate channels, we observe either full detection of all channel clients or complete lack of detection. This is due the legitimate IRC communication is done through the exchange of broadcast messages and, thus, if any observer is situated on the path of the IRC traffic, it finds all the clients of the channel.

#Sensor	#Irc-bot	#Irc-1	#Irc-2
sensor_sas17	97,91%	100,00%	100,00%
sensor_tas0	95,82%	100,00%	100,00%
sensor_tas4	26,82%	100,00%	100,00%
sensor_tas2	26,00%	100,00%	100,00%
sensor_sas1	15,00%	100,00%	100,00%
sensor_sas18	7,27%	0,00%	0,00%
sensor_sas26	5,45%	100,00%	0,00%
sensor_sas11	5,45%	0,00%	0,00%
sensor_tas8	5,27%	100,00%	0,00%
sensor_tas5	5,27%	0,00%	0,00%
sensor_sas20	5,09%	100,00%	0,00%
sensor_sas13	5,00%	0,00%	0,00%

Table 2. Relationship Metrics of IRC Channels

We see strong differentiation of observed metrics, depending from the observer network position. This is due to peculiarities of communication of botnet clients in the IRC control channel. Instead of using broadcast messages directed to all participants in the channel, bots exchange information only with a small number of nodes, belonging to botnet masters. We observe almost complete detection of botnet control channel for two routers in table 2. Analysis of the topology of the simulated network shows that the segment sas17 (sensor_sas17) has an IRC server node. The segment tas0, located in proximity to the segment sas17, is a transit for traffic between the IRC server and the most of IRC bots.

Thus, we can suppose that a defense mechanism, fulfilled on a small number of routers which are transit for the main IRC traffic, can be as effective as the defense mechanism installed in more number of routers. We can also assume that a defense mechanism, having a small covering of the protected network, generally will not be efficient, because only a small part of IRC control traffic passes the vast majority of routers.

IRC traffic monitoring and synchronization metric calculation. In these experiments the IRC traffic is monitored in different network locations. Based on monitoring results, the synchronization metrics are calculated. Let us consider the synchronization metrics determined by monitoring the traffic on the core router of network segment tas0 (Fig.14).

From 200 seconds of simulation time, every 100 seconds we can observe sharp spikes of traffic volume related to the botnet control IRC channel. These bursts are caused by response messages from zombies on a request from the botnet master.

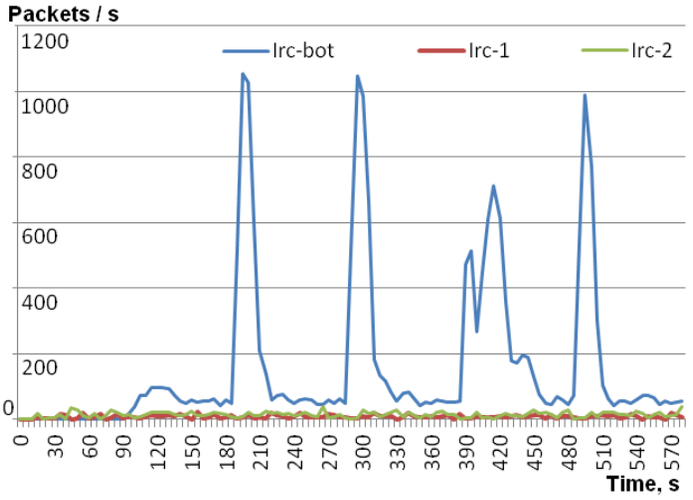


Figure 14. Synchronization Metrics for tas0

Network segment tas0 is located in proximity from the network segment which includes IRC-server. Thus, a significant part of IRC control traffic is transmitted through the router of network segment tas0. For this reason, the bursts of control channel traffic are markedly expressed against the traffic of legitimate communication.

Traffic on a network segment router sas13 was measured (Fig.15) to evaluate the impact of the proximity of the observation point from the IRC server on the severity of bursts of control traffic (and thus on the discernibility of synchronization metric).

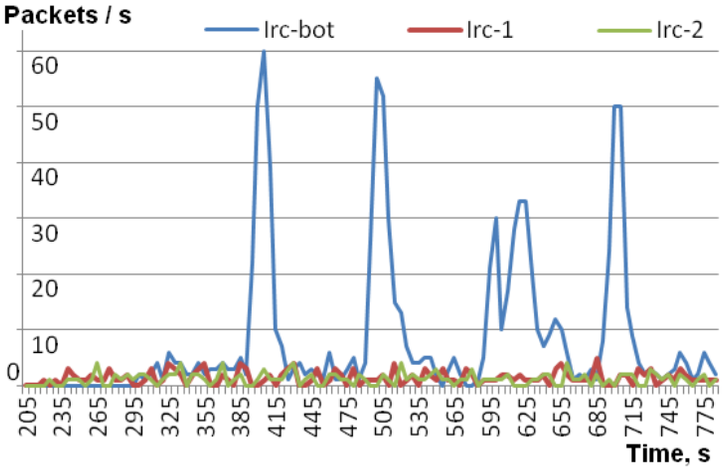


Figure 15. Synchronization Metric for sas13

Traffic measurements show a general decrease of traffic level in the observation point sas13, as well as a good visibility of traffic spikes on the core router of this network segment. Thus, the results of experiments demonstrate the applicability of synchronization metric to detect the IRC control traffic.

IRC traffic filtering based on relationship metric. This filtering method is based on the assumption that the IRC channels with a very large number of clients are anomalous. We carried out a series of experiments where the relationship metric was used for different configurations of filtering components and different critical levels of relationship. It was shown that the efficiency of IRC traffic detection and filtering, based on relationship metric, increases sharply when the routers, which are transit for IRC control traffic, are fully covered by filtering components.

IRC traffic filtering based on synchronization metric. This filtering method is based on the assumption that the short synchronous messaging in a single IRC channel is anomalous. The observed synchronization metric is calculated as the number of IRC packets, passing through the observation point, for a fixed period of time. In the experiments fulfilled, the filtering criterion is a fivefold increase in traffic for 20 seconds followed by a return to its original value. The results of experiments allow concluding about low quality of the method in the current configuration, since false positive rate has a rather high value.

c. DDoS attacks and defense against them

The module of DDoS attacks is parameterized for experiments described as follows: type of attack – SYN flooding; flooding frequency – 10, 30 or 60 packets per second; total number of packets to send by a single host – 1000. Attacks take some Web server as a target, so TCP port 80 is the destination port during attack stage. Spoofing of source IP address is fulfilled in some experiments. We adopt address range that is subset of address range of the whole network to implement spoofing.

Two kinds of defense mechanisms are considered in the description of experimental results: SAVE and SIM.

At the 400th second of model time, bot master initiates the beginning of the attack stage by sending broadcasting network message through the IRC server. This message is transmitted to all "zombie" hosts involved into the botnet. Other data, enveloped into this message, contains specification of IP address and port of the target host. Every zombie, received this message, is able to extract such information and use it as a parameter of own DDoS related activity.

Fig.16 shows the number of packets targeted to the victim host relative to the model time while SAVE method is active. Different portion of routers is used as hosts for defense method deployment. The metrics for 30%, 50% and 100% of router coverage are shown.

The false positive rate, false negative rate and correct detection rate relative to model time are shown in Fig.17. Such metrics are observed for the traffic passing through the core router of the network which victim host belongs to. The only SIM method was enabled. Fig.18 depicts the relative estimation of filtered legitimate traffic against the model time. Relative estimation of the traffic is calculated as a ratio of the number of filtered packets to the number of all packets passed through the SIM defense mechanism. Three cases for different protection coverage (30%, 50% and 100%) were considered.

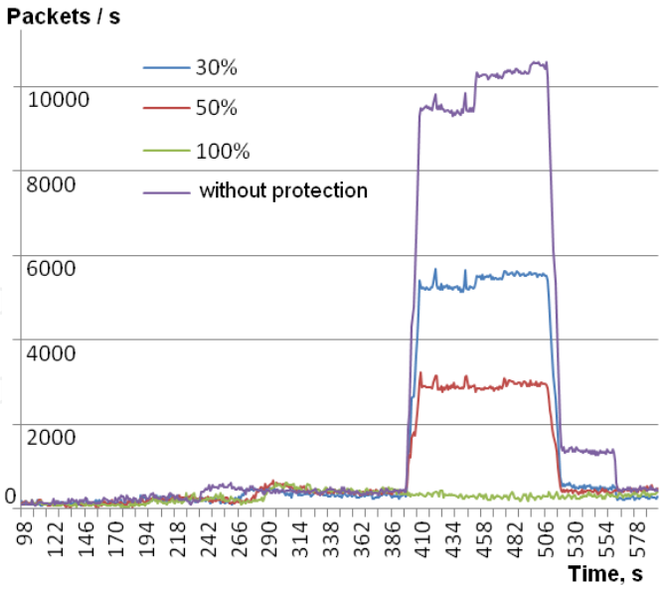


Figure 16. Number of Incoming Packets on Target Host when SAVE Method is enabled

The SIM defense mechanism reveals a high level of TP and a very low level of FN in all cases (Fig.16). A tiny spike of FN observed only at the beginning of attack stage (Fig.17). The level of FP increases gradually due to the fact of continual increment of dropped packets with unspecified IP-addresses, since the beginning of attack stage. It is noted that the ratio of filtered legitimate packets can reach up to 30-40% of legitimate traffic (Fig.18).

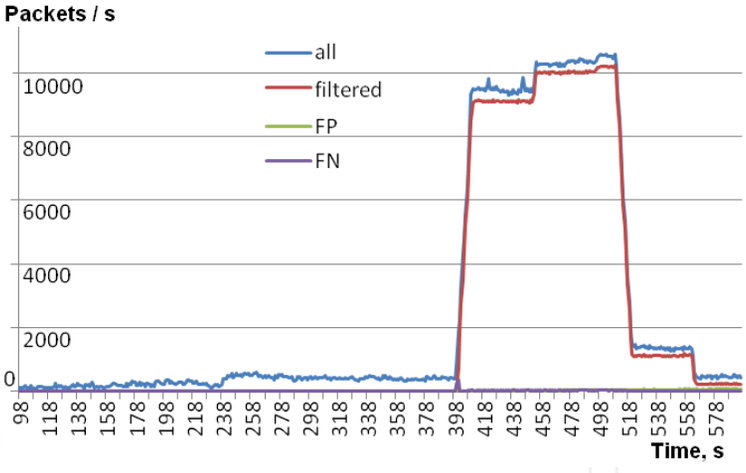


Figure 17. Main metrics of SIM

d. Comparison of with the Results of Emulation

To verify the developed simulation models, we emulated the functioning of small networks consisting of many nodes on real computers combined to a network using Oracle VM Virtual Box. On emulated computers the typical software was installed, and the work of legitimate users and malefactors was imitated. To emulate the botnet such hosts as “master”, “control center” and “vulnerable computers” were selected. Furthermore, the software for monitoring of network traffic was installed on the computers.

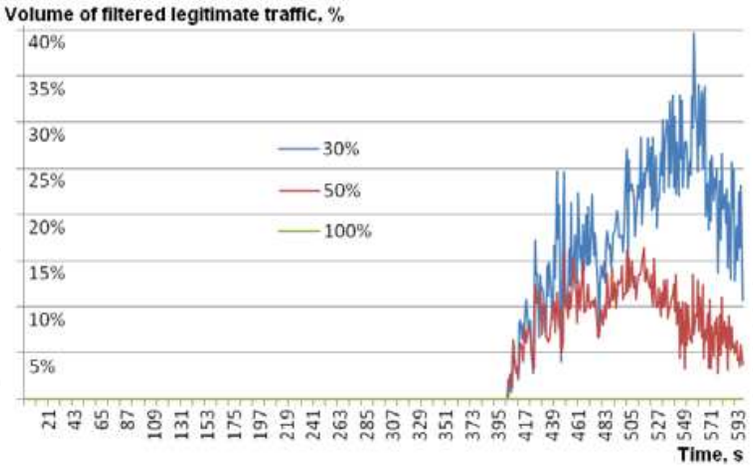


Figure 18. Volume of filtered legitimate traffic

Using the developed network testbed, we compared the results obtained on the basis of simulation models with the results of emulation. In case of discrepancies in the results the corresponding simulation models were corrected.

To test the adequacy of the simulation models, the network consisting of 20 virtual nodes was built.

The examples of parameters which were evaluated for the real network are as follows: packet delay, packet loss, the rate of infection of vulnerable hosts, the number of bots participating in the attack, the load of the victim node during DDoS-attack, etc.

In the normal state of the network, the packet delay for legitimate packets to reach the victim node was from 3 to 7 milliseconds, the packet loss was less than 1%. When we emulated a worm spreading, the rate of infection of 18 nodes was from 2.2 to 2.4 s. When performing DDoS-attack, the victim node received from 174 to 179 packets per second, and packet loss increased to 4%. In the case of simulation with the same parameters, the packet delay was 5 milliseconds with the dispersion - 2 milliseconds and packet loss - 1%. The rate of infection of 18 nodes was about 2.4 s. The number of packets used for DDoS-attacks was about 180.

7. Conclusion

The paper suggested a common approach to investigative modelling and packet-level simulation of botnets and defense mechanisms against them in the Internet. We proposed a generalized architecture of simulation environment aiming to analyze botnets and defense mechanisms. On the base of this architecture we designed and implemented a multilevel software simulation environment. This environment includes the system of discrete event simulation (OMNeT++), the component of networks and network protocols simulation (based on INET Framework library), the component of realistic networks simulation (using the library ReaSE) and BOTNET Foundation Classes library consisting of the models of network applications related to botnets and defense against them.

The experiments investigated botnet actions and protection mechanisms on stages of botnet propagation, botnet management and control (reconfiguration and preparation to attacks), and attack execution. We analyzed several techniques, including Virus Throttling and Failed Connection, to protect from botnet on the propagation stage. Botnet propagation was performed via network worm spreading. We researched techniques of IRC-oriented botnet detection to counteract botnets on the management and control stage. These techniques are based on the "Relationship" metric of particular IRC-channels, metric of the distribution of response time to the broadcasting request ("Response") and the metric of botnet group behavior synchronization ("Synchronization"). We also analyzed techniques, which work on the different stages of defense against DDoS attacks. These techniques include SAVE (Source Address Validity Enforcement Protocol), SIM (Source IP Address Monitoring) and Hop-count filtering.

The purpose of this paper is to provide an environment for simulation of computer networks, botnet attacks and defense mechanisms against them. This simulation environment allows investigating various processes in computer networks - the performance of communication channels and servers, the operation of computer network nodes during different attacks on them, the effect of protection mechanisms on the computer network, the best strategies for location and implementation of protection mechanisms.

The developed simulation environment allows changing the main parameters for conducting experiments. These parameters can be adjusted to simulate different types of worms, DDoS attacks, command centers' operation, as well as various protection mechanisms with a wide range of values.

By changing the values of the parameters used to model the life cycle of botnets and protection mechanisms against them, we can generate different types of botnets. For example, it is possible to simulate the spread of botnets for a few milliseconds or the case of their blocking at the first stage of the operation.

The experiments fulfilled were based on typical values of the parameters that can demonstrate the overall dynamics of the development and operation of botnets and the ability to implement different protection mechanisms. The conclusions derived from the simulation results are generalizable to other cases where values of these parameters are outside the range or different from those investigated.

The developed environment allows building the models based on the real network topologies with highly detailed units included in the network to provide the high fidelity of the models.

We suppose that suggested approach can be used to investigate operation of different types of botnets, to evaluate effectiveness of defense mechanisms against botnets and other network attacks, and to choose optimal configurations of such mechanisms.

Future research is connected with the analysis of effectiveness of botnet operation and defense mechanisms, and improvement of the implemented simulation environment. One of

the main tasks of our current and future research is to improve the scalability and fidelity of the simulation. We are in the process of experimenting with parallel versions of the simulation environment and developing a simulation and emulation tesbed, which combines a hierarchy of macro and micro level analytical and simulation models of botnets and botnet defense (analytical, packet-based, emulation-based) and real small-sized networks.

Author details

Igor Kotenko, Alexey Konovalov and Andrey Shorov
Laboratory of Computer Security Problems, St.-Petersburg Institute for Informatics and Automation of Russian Academy of Sciences, St. Petersburg, Russia

Acknowledgement

This research is being supported by grants of the Russian Foundation of Basic Research (project 10-01-00826), the Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (2.2), State contract #11.519.11.4008 and partly funded by the EU as part of the SecFutur and MASSIF projects.

8. References

[1] Akiyama M, Kawamoto T, Shimamura M, Yokoyama T, Kadobayashi Y, Yamaguchi S. (2007) A proposal of metrics for botnet detection based on its cooperative behavior, SAINT Workshops, pp. 82-82.
[2] Bailey M, Cooke E, Jahanian F, Xu Y, Karir M (2009) A Survey of Botnet Technology and Defenses, Cybersecurity Applications Technology Conference for Homeland Security,.
[3] Binkley JR, Singh S (2006) An algorithm for anomaly-based botnet detection", Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet, Vol.2.
[4] Chen S, Tang Y (2004) Slowing Down Internet Worms, Proceedings of the 24th International Conference on Distributed Computing Systems.
[5] Dagon D, Zou C, Lee W (2006) Modeling botnet propagation using time zones, Proc. 13th Annual Network and Distributed System Security Symposium. San Diego, CA.
[6] Feily M, Shahrestani A, Ramadass S (2009) A Survey of Botnet and Botnet Detection", Third International Conference on Emerging Security Information Systems and Technologies.
[7] Gamer T, Mayer C (2009) Large-scale Evaluation of Distributed Attack Detection, 2nd International Workshop on OMNeT++.
[8] Grizzard JB, Sharma V, Nunnery C, Kang BB, Dagon D (2007) Peer-to-Peer Botnets: Overview and Case Study.

[9] Huang Z, Zeng X, Liu Y (2010) Detecting and blocking P2P botnets through contact tracing chains, International Journal of Internet Protocol Technology archive, Vol.5, Issue 1/2.
[10] Hyunsang C, Hanwoo L, Heejo L, Hyogon K (2007) Botnet Detection by Monitoring Group Activities in DNS Traffic, 7th IEEE International Conference on Computer and Information Technology CIT, pp.715-720.
[11] The INET Framework is an open-source communication networks simulation package for the OMNeT++ simulation environment. Available: http://inet.omnetpp.org/. Accessed 2012 Marth 24.
[12] Kotenko I (2010) Agent-Based Modelling and Simulation of Network Cyber-Attacks and Cooperative Defence Mechanisms", Discrete Event Simulations, Sciyo, pp.223-246.
[13] Kotenko I, Konovalov A, Shorov A (2010) Agent-based Modeling and Simulation of Botnets and Botnet Defense", Conference on Cyber Conflict. CCD COE Publications. Tallinn, Estonia, pp.21-44.
[14] Krishnaswamy J (2009) Wormulator: Simulator for Rapidly Spreading Malware, Master's Projects.
[15] Kugisaki Y, Kasahara Y, Hori Y, Sakurai K (2007) Bot detection based on traffic analysis, Proceedings of the International Conference on Intelligent Pervasive Computing, pp.303-306.
[16] Li L, Alderson D, Willinger W, Doyle J (2004) A first-principles approach to understanding the internet router-level topology", ACM SIGCOMM Computer Communication Review.
[17] Li J, Mirkovic J, Wang M, Reither P, Zhang L (2002) Save: Source address validity enforcement protocol", Proceedings of IEEE INFOCOM, pp.1557-1566.
[18] Mao C, Chen Y, Huang S, Lee H (2009) IRC-Botnet Network Behavior Detection in Command and Control Phase Based on Sequential Temporal Analysis, Proceedings of the 19th Cryptology and Information Security Conference.
[19] Mazzariello C (2008) IRC traffic analysis for botnet detection, Proceedings of Fourth International Conference on Information Assurance and Security.
[20] Nagaonkar V, Mchugh J (2008) Detecting stealthy scans and scanning patterns using threshold random walk", Dalhousie University.
[21] Naseem F, Shafqat M, Sabir U, Shahzad A (2010) A Survey of Botnet Technology and Detection, International Journal of Video & Image Processing and Network Security, Vol.10, No. 1.
[22] Owezarski P, Larrieu N (2004) A trace based method for realistic simulation, Communications, 2004 IEEE International Conference.
[23] Peng T, Leckie C, Ramamohanarao K (2004) Proactively Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring, Lecture Notes in Computer Science, Vol.3042, pp.771-782.
[24] ReaSE - Realistic Simulation Environments for OMNeT++. Available: https://i72projekte.tm.uka.de/trac/ReaSE. Accessed 2012 Marth 24.

[25] Riley G, Sharif M, Lee W (2004) Simulating internet worms, Proceedings of the 12th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), pp.268-274.
[26] Ruitenbeek EV, Sanders WH (2008) Modeling peer-to-peer botnets, Proceeding of 5th International Conference on Quantitative Evaluation of Systems, pp. 307-316.
[27] Schuchard M, Mohaisen A, Kune D, Hopper N, Kim Y, Vasserman E (2010) Losing control of the internet: using the data plane to attack the control plane, Proceedings of the 17th ACM conference on Computer and communications security, pp.726-728.
[28] Sen S, Spatscheck O, Wang D (2004) Accurate, scalable in-network identification of p2p traffc using application signatures, Proceedings of the 13th international conference on World Wide Web, pp. 512-521.
[29] Simmonds R, Bradford R, Unger B (2000) Applying parallel discrete event simulation to network emulation, Proceedings of the fourteenth workshop on Parallel and distributed simulation.
[30] Suvatne A. Improved Worm Simulator and Simulations. Master's Projects, 2010.
[31] Varga A. (2010) OMNeT++. Chapter in the book "Modeling and Tools for Network Simulation", Wehrle, Klaus; Günes, Mesut; Gross, James (Eds.) Springer Verlag.
[32] Villamarín-Salomón R, Brustoloni JC (2009) Bayesian bot detection based on DNS traffic similarity, Proceeding SAC '09 Proceedings of the 2009 ACM symposium on Applied Computing.
[33] Vishwanath KV, Vahdat A (2006) Realistic and responsive network traffic generation, Proceedings of the Conference on Applications, technologies, architectures, and protocols for computer communications.
[34] Wang P, Sparks S, Zou CC (2007) An advanced hybrid peer-to-peer botnet, Proceedings of the First Workshop on Hot Topics in Understanding Botnets.
[35] Wang H, Zhang D, Shin K (2002) Detecting SYN flooding attacks, Proceedings of IEEE INFOCOM, pp.1530–1539.
[36] Wehrle K, Gunes M, Gross J (2010) Modeling and Tools for Network Simulation, Springer-Verlag.
[37] Williamson M. (2002) Throttling Viruses: Restricting propagation to defeat malicious mobile code", Proceedings of ACSAC Security Conference, pp.61–68.
[38] Zhou S, Zhang G, Zhang G, Zhuge Zh (2006) Towards a Precise and Complete Internet Topology Generator", Proceedings of International Conference Communications.