World Research Journal of Computer Architecture
ISSN: 2278-8514 & E-ISSN: 2278-8522, Volume 1, Issue 1, 2012, pp.-06-10.
Available online at http://www.bioinfo.in/contents.php?id=97

APPLICATION OF NS2 TO OVERCOME COMPUTER NETWORKS ATTACKS

SURABHI AGRAWAL AND RIMA LINGAWAR

CSE, J.D.I.E.T., Yavatmal, MS, India.
*Corresponding Author: Email- surabhi.agrawal34@yahoo.com and rima.lingawar@yahoo.com

Received: February 28, 2012; Accepted: May 09, 2012

Abstract- Experiment system for computer network is possible to construct using NS2. Security is an essential in order to solve the realistic problems in the experiments of computer networks. Demonstration and requirement in mobile ad hoc network (MANETs). Compared to wired networks, MANETs are more vulnerable to security attacks due to the lack of a trusted centralized authority and limited resources. Attacks on ad hoc networks can be classified as passive and active attacks, depending on whether the normal operation of the network is disrupted or not. There exist some best network simulation tools in the computer network research area, such as NS2 and OPNET, which provide very powerful simulation functions for network protocols. One main challenge in design of these networks is their vulnerability to security attacks. Study of the attacks that are possible in Ad-Hoc networks faces and means to model the same in Network Simulator-2. We use well-established techniques to gain knowledge about the network topology and use this knowledge to perform plausibility checks of the routing information propagated by the nodes in the network.
Keywords- Survey, Security attacks, Mobile ad hoc network (MANET).

Citation: Surabhi Agrawal and Rima Lingawar (2012) Application of NS2 to overcome Computer Networks Attacks. World Research Journal of Computer Architecture, ISSN: 2278-8514 & E-ISSN: 2278-8522, Volume 1, Issue 1, pp.-06-10.

Introduction

There exist some outstanding network simulation tools in the computer network research area, such as NS2 and OPNET, which provide very powerful simulation functions for network protocols, and through which it is very convenient to create the running environment for various protocols and it is very easy to display the protocol behavior visually. Nevertheless these simulation tools are very hard to install and utilize. At present they are used by computer network researchers to study, extend and develop new protocols.

By using NS2-based demonstration and experiment system for computer network courses, which builds a bridge between network the simulation software NS2 and the students who want to learn computer network courses. Through this system, the students not only can leave out the difficulty to learn the network simulation software NS2, but also can make use of its powerful simulation functions to carry out network simulation experiments, and consequently understand the complex behavior of network protocols more comprehensively. This demonstration and experiment system can be applied to the education of computer network related courses.

The remainder of the article is organized as follows. The network is introduced briefly followed by network simulation software NS2 is introduction in the next section. Conclusions are drawn in the last section.

NS2 overview

Network simulation is a kind of technology that simulates the network behavior through mathematical modeling and statistical analysis method and then obtains the specific parameters which reflect the characteristics of the network. NS2 is one of the most famous network simulation software which was developed by LBNL network research group at UC Berkeley in the USA. It is primarily useful for simulating local and wide area networks, no matter what are wired or wireless networks. It has great value for network researchers, especially for the designers of new network protocols.

NS2 is an object-oriented, discrete event driven network simulation tool. One major component of NS2 is the event scheduler. An event in NS2 is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS2, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, which usually are the ones who issued the events, and let them do the appropriate action associated with packet pointed by the event. Network components communicate with one the other passing packets, however this does not consume actual simulation time. All the network components that need to spend some simulation time handling a packet use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet. The other use of an event scheduler is timer. Timers use event schedulers in a similar manner that delay does. The only difference is that timer measures a time value associated with a packet and does an appropriate action related to that packet after a certain time goes by, and does not simulate a delay.
NS2 is designed to simulate variety of IP networks. It covers a very large number of applications, network types, network elements and traffic models which are called simulated objects. It implements network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS2 also implements multicasting and some of the MAC layer protocols for LAN simulations. Besides, NS2 supports the simulation of various new technologies such as GRPS, mobile IPv6, RSRV, MPLS and Ah Hoc networks. The NS2 project is now a part of the VINT project that develops tools for simulation results display, analysis and converters that convert network topologies generated by wellknown generators to NS formats.
NS2 simulator is based on two languages: an object oriented simulator, written in C++, and a OTcl (an object oriented extension of Tcl), used to execute user's command scripts. For efficiency reason, NS2 separates the data path implementation from control path implementations. In order to reduce packet and event processing time, the event scheduler and the basic network component objects in the data path are written and compiled using C++. These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects and makes the control functions and configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object. In this way, the controls of the C++ objects are given to OTcl. It is also possible to add member functions and variables to a C++ linked OTcl object. The objects in C++ that do not need to be controlled in a simulation or internally used by the other object do not need to be linked to OTcl. Likewise, an object can be entirely implemented in OTcl.
In order to observe and analyze the simulation result intuitively, NS2 provides a graphical simulation display tool called Network Animator (NAM). NAM has a nice graphical user interface similar to that of a CD player, and also has a display speed controller. Furthermore, it can graphically present information such as throughput and number of packet drops at each link, although the graphical information cannot be used for accurate simulation analysis. The design theory behind NAM was to create an animator that is able to read large animation data sets and be extensible enough so that it could be used indifferent network visualization situations. Under this constraint NAM was designed to read simple animation event commands from a large trace file.
The general work process of network simulation using NS2 is summarized as shown in Fig. 1.

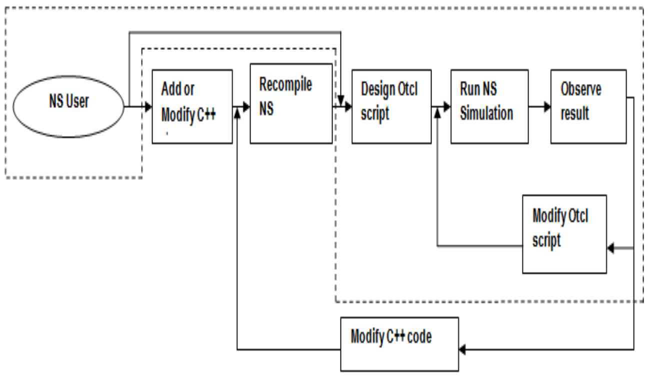


Fig. 1- General Work Process of Network Simulation

Hierarchy of an Object in NS-2

TclObject is the base class for most of the other classes in the interpreted and compiled hierarchies. Every object in the class TclObject is created by the user from within the interpreter.

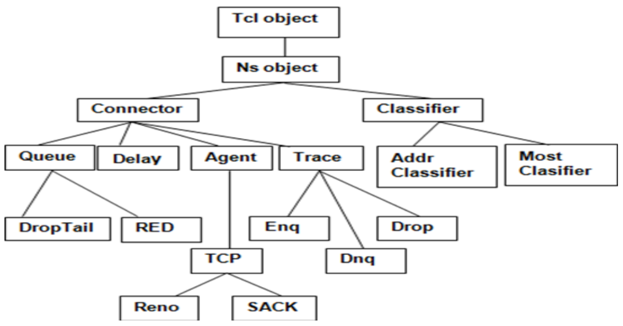


Fig 2- Hierarchy of an object in NS-2

NS Code Hierarchy

The diagram given below gives a brief description on how the C++ and Otcl code is organized in NS-2.

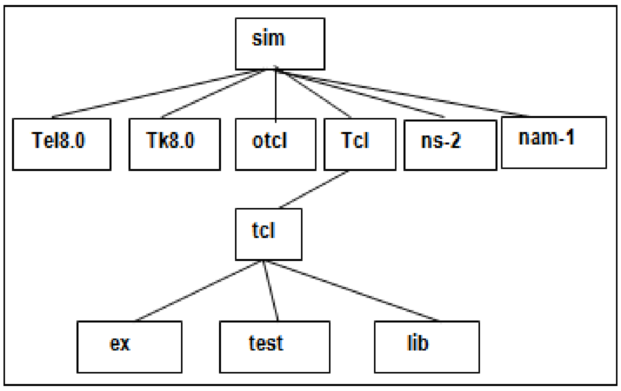


Fig. 3- NS Code Hierarchy

Classification of Attacks

The attacks can be classified in different ways. They can be based on the sources of the attacks (internal attack, external attack), or on the methods through which the attackers acquire control (e.g. Fake error, False reply) or they can be classified on the basis of security service it achieves to attack. The attacks are classified into active and passive at the first level and then at the second level I classify the attacks on the basis of which security service is being attacked e.g. confidentiality, Integrity, authentication etc. and finally I specify how the attack objective is achieved.

Passive Attack

A passive attack can be described as an attack in which an attacker does not actively perform an attack, in the sense that it actively does not initiate malicious actions to cheat other hosts. One example of this kind of an attack is Eaves dropping. The attacker launches this type of an attack to determine the network topology by looking at the packets that are exchanged between neighboring nodes. Detection of passive attack is very difficult since the operation of the network itself doesn't get affected. One of the solutions to the problem is to use powerful encryption mechanism to encrypt the data being transmitted, by making it impossible for the attacker to get useful information from the data overhead. The attack initially looks innocuous, but can be extremely dangerous when combined with an active attack.

Active Attack

An active attack attempts to alter or destroy the data being exchanged in the network there by disrupting the normal functioning of the network. An active attack is the one in which the malicious host generates an active attack by introducing false information into an Ad Hoc network. It confuses routing procedures and degrades network performance. Active attacks can be internal or external. External attacks are carried out by nodes that do not belong to the network. Internal attacks are from compromised nodes that are part of the network. Since the attacker is already part of the network, internal attacks are more severe and hard to detect than external attacks. Active attacks, whether carried out by an external advisory or an internal compromised node involves actions such as impersonation, modification, fabrication and replication.

Network Layer Attacks in NS2 are as follows

Gray Hole Attack

The gray hole attack has two phases. In the first phase, a malicious node exploits the AODV protocol to advertise itself as having a valid route to a destination node, with the intention of intercepting packets, even though the route is spurious. In the second phase, the node drops the intercepted packets with a certain probability. This attack is more difficult to detect than the black hole attack where the malicious node drops the received data packets with certainly. A gray hole may exhibit its malicious behavior in different ways. It may drop packets coming from (or destined to) certain specific node (s) in the network while forwarding all the packets for other nodes. Another type of gray hole node may behave maliciously for some time duration by dropping packets but may switch to normal behavior later. A gray hole may also exhibit a behavior which is a combination of the above two, thereby making its detection even more difficult.

Wormhole Attack

An attacker records packets at one location in the network and tunnels them to another location. Routing can be disrupted when routing control messages are tunneled. This tunnel between two colluding attackers is referred as a wormhole. Wormhole attacks are severe threats to MANET routing protocols. For example, when a wormhole attack is used against an on-demand routing protocol such as DSR or AODV, the attack could prevent the discovery of any routes other than through the wormhole.

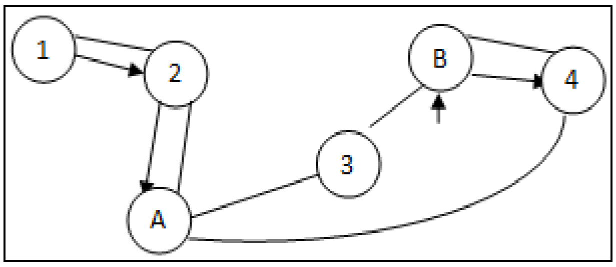


Fig. 4- Wormhole Attack

Blackhole Attack

The blackhole attack has two properties. First, the node exploits the mobile ad hoc routing protocol, such as AODV, to advertise itself as having a valid route to a destination node, even though the route is spurious, with the intention of intercepting packets. Second, the attacker consumes the intercepted packets without any forwarding. However, the attacker runs the risk that neighboring nodes will monitor and expose the ongoing attacks. There is a more subtle form of these attacks when an attacker selectively forwards packets. An attacker suppresses or modifies packets originating from some nodes, while leaving the data from the other nodes unaffected, which limits the suspicion of its wrongdoing.

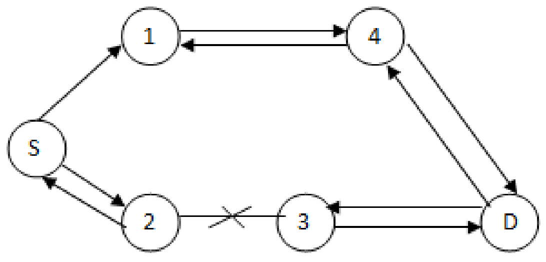


Fig. 5- Blackhole Attack

Routing Attacks

There are several types of attacks mounted on the routing protocol which are aimed at disrupting the operation of the network. Various attacks on the routing protocol are described briefly below:

1. Routing Table Overflow
2. Route Cache Poisoning

System Construction

A NS2-based demonstration and experiment system is made up of two parts, one of which is the demonstration system for eg. teachers to demonstrate in class, and the other is the experiment system for eg. students to do experiment in the laboratory or school. In this system we do not need to modify the protocols that NS2 has already implemented, need to add new protocols neither. Therefore, we do not need to add or modify the C++ code in NS, need to recompile NS neither. The NS simulation process using by this demonstration and experiment system is described in dotted line frame part of Fig. 1. The process only includes designing or modifying Otcl script, running NS simulation and observing the results.

The Demonstration System

This demonstration system is used for teachers to demonstrate in class. The NS process using by this demonstration system is described in Fig. 2. It is made up of two processes, one of which is the experiment database developing process, and the other is the demonstrating process in class. The experiment database developing process is described above the dotted line in Fig. 2. The purpose of this process is to prepare the material for classroom demonstration and as a result to form the experiment database. The database can be modified and cited by developers or teachers conveniently when needed. The steps to develop the experiment database are as follows. Firstly, the developers design the Otcl script according to the contents of the courses. For example, in order to demonstrate the slow start mechanism in TCP congestion control, we design two pieces of Otcl script. One is named *WithSlowStart.tcl*, and the other is named *WithoutSlowStart.tcl*. Both of them use the similar simulation environment such as the topology and the traffic type.The difference between them is that the former uses TCP but the latter uses TCP without slow start with slow start mechanism, due to paper length limitation. Next, run NS simulation on the designed Otcl script, then the corresponding NAM trace file will be created automatically. In the above example, we run NS simulation on *WithSlowStart.tcl* and *WithoutSlowStart.tcl* respectively, and then we can get two corresponding NAM trace files. They are named *WithSlowStart.nam* and *WithoutSlowStart.nam* respectively. Lastly, observe and analyze the experiment results using the NAM tool. Also in the above example, we open *WithSlowStart.nam* and *WithoutSlowStart.nam* in the NAM tool respectively and compare the two experiment results. The emphasis is whether the results reflect the difference between TCP with slow start mechanism and without slow start mechanism. The results also show the changing process of the congestion window size, from which students can understand the mechanisms in TCP congestion control comprehensively. If the results are satisfied with the requirements of the classroom demonstration, add the Otcl script and NAM trace file into the Otcl script database and NAM trace file database respectively.

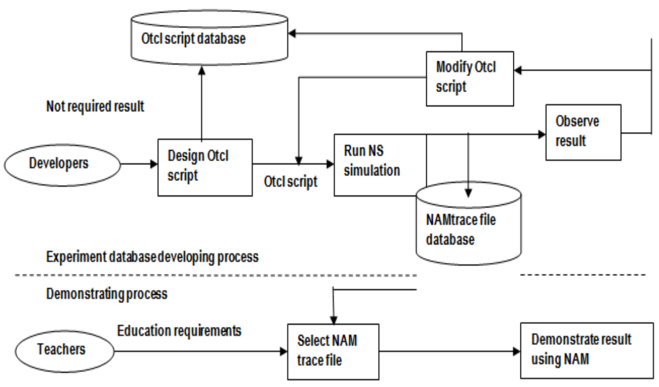
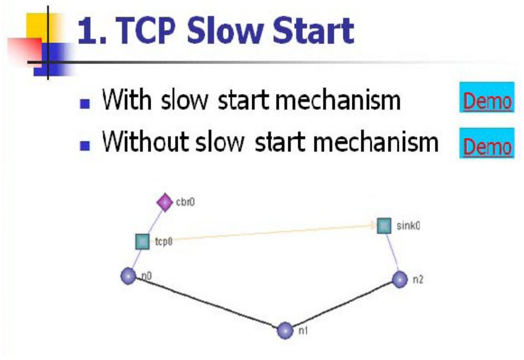


Fig. 6- Wormhole Attack

Otherwise, modify the Otcl script and repeat the above process, i.e. running NS simulation, observing the results and making the decision to add the script and file to database or not. After the development work is finished, the experiment database creates, which is made up of the Otcl script database and the NAM trace file database. The experiment database can be used for classroom demonstration or be modified as required. The demonstrating process in class is described below the dotted line in Fig. 2. The detailed steps can be described as follows. First of all, the teachers select the appropriate material from the experiment database according to the educational purpose. For instance in order to explain the slow start mechanism in TCP congestion control, the NAM trace files named *WithSlowStart.nam* and *WithoutSlowStart.nam* in above example can be choose. Note that here only NAM trace file is needed. The Otcl script file is not needed at all. In the next place, import the selected NAM trace file into Microsoft Powerpoint document. We have solved the problem to open NAM trace file in Powerpoint document, hence the teachers can insert the needed NAM trace file into the Powerpoint document which they want to use in class very conveniently. For example, in order to demonstrate the TCP slow start mechanism, the Powerpoint document page as shown in Fig. 3 can be designed. The ultimate step is to demonstrate the results in class. In above instance, teachers only need to press the "Demo" button in the Powerpoint document page, then the corresponding NAM trace file can be opened using the NAM tool automatically. The work process of the slow start mechanism can be demonstrated to the students.



1. TCP Slow Start

- With slow start mechanism Demo
- Without slow start mechanism Demo

Conclusion

The security of the Ad Hoc network routing protocols is an open problem and deserves more research work in this area. In this, I have analyzed the security threats an ad hoc network faces and presented mean to model these attacks in Network Simulator-2. On one hand, the security-sensitive applications of ad hoc networks require high degree of security; on the other hand, ad hoc networks are inherently vulnerable to security attacks. However the extent of damage of a particular attack cannot be fully understood without actually implementing the attack in a simulator such as NS-2. An attack would seem extremely deadly at the face of it, but the mechanisms within the underlying protocol might undermine the extent of the attack. For e.g. in a black hole attack if the attacker nodes continuously drops packet from the source node, the source night a initiate a new route request and establish the path to the destination through a different node which is not an attacker node.

References

[1] Xu L.M., Pang B. and Zhao G.Y. (2003) *NS and network simulation*, Posts & Telecom Press, Beijing.
[2] NS2 Main Page [Online], *http://nsnam.isi.edu/nsnam/index.php/Main_Page*.
[3] NS Bench project description, *http://www.mnlab.cs.depaul.edu/projects/nsbench/main.htm*.
[4] Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6).
[5] *http://www.isi.edu/nsnam/ns/*.