Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 1, pp. 114–122, DOI 10.12694/scpe.v26i1.3670

CYBERGUARD: A FORTIFIED MESSAGE AUTHENTICATION PROTOCOL WITH DIGITAL SIGNATURES IN NS-2 SIMULATION ENVIRONMENT

PRASHANT KUMAR*, DIMPLE SETHI†, GULSHAN SHRIVASTAVA‡, AND NEERAJ SRIVASTAVA§

Abstract. Delving into the dynamic realm of communication networks, the Network Simulator-2 (NS-2) emerges as a versatile, object-oriented, and event-driven mirror, analysing the intricacies of computer network. NS-2, an open-source, is a strong tool for protocol innovation that supports numerous routing protocols designed for both wired and wireless networks in addition to the TCP/IP protocol suite. However, the security frontier has remained a latent challenge amidst its diverse protocol array. This research defiantly explores the security void in NS-2 and presents a novel solution by adding a state-of-the-art security module. Elevating NS-2's prowess, this module seamlessly integrates message integrity and sender authentication features, injecting a much-needed security boost. We meticulously detail the inner workings of the security modules, the innovative processes deployed, and the simulation and implementation intricacies within NS-2. The limelight is on the propagation of sender authentication protocols and the embodiment of message integrity in wired communication networks. Network Animator steps onto the stage to make the simulation journey more vivid, providing an engaging visual narrative. At its core, this module strives to democratize the integration of Digital Signature features, carving a path toward a more secure NS-2 landscape. This paper augments NS-2's resilience and charts new territories in the dynamic intersection of communication networks and cutting-edge security protocols.

Key words: Network Simulator-2, Decryption/Encryption, Integrity of Message, Sender Authentication, Asymmetric Key Cryptography, Network Layer Security, RSA Algorithm

1. Introduction. Network Simulator-2 is an event-driven open-source simulator developed primarily for research [15, 11, 8]. Experimenters can add additional modules and functions to NS-2 to meet their requirements. NS-2 comprises C++ and object-oriented Tool Command Language (OTCL) [7, 6]. C++ describes the inner workings (backend) of the simulation objects, whereas oTcl starts the simulation by building and configuring the objects and scheduling discrete events (frontend). NS-2 supports protocols from several levels, such as (the most recent version of NS-2 is version 2.35). Network Simulator-2 (NS-2) is a powerful, event-driven, open-source simulator expressly crafted for research endeavors. It offers a versatile platform wherein experimenters can seamlessly integrate new modules and functionalities to tailor the simulation to their specific requirements. NS-2 operates in two languages: C++ defines the internal mechanisms (backend) of simulation objects, while the object-oriented Tool Command Language (OTCL) orchestrates simulation setup by assembling and configuring objects, as well as scheduling discrete events (frontend). The latest version, NS-2 version 2.35, encompasses a comprehensive suite of standard protocols across various layers. From CSMA/CD at the data link layer to FTP, TELNET, DNS, and HTTP at the application layer, and encompassing both multicast and unicast routing protocols for wired networks, along with DSDV, DSR, and AODV for wireless networks, NS-2 has become a repository of advanced protocols embraced by the research community.

This paper introduces an innovative approach to enhance NS-2's capabilities by incorporating a security module. The proposed module facilitates the integration of message integrity and sender authentication functions. A novel packet format is constructed using a class derived from an NS-2 built-in class to instantiate a new protocol at the IP layer. This new class includes encryption/decryption functions for the data field in the data packet, ensuring sender authentication. The integrity of data during communication is confirmed using

*SCSET, Bennett University (The Times Group), Plot No 8-11, Techzone 2, Greater Noida, Uttar Pradesh 201310, India (prashant.mnnit10@gmail.com).
†SCSET, Bennett University (The Times Group), Plot No 8-11, Techzone 2, Greater Noida, Uttar Pradesh 201310, India (dimple9203@gmail.com)).
‡SCSET, Bennett University (The Times Group), Plot No 8-11, Techzone 2, Greater Noida, Uttar Pradesh 201310, India (Corresponding author, gulshanstv@gmail.com).
§Department of Mathematics, Agra college, Agra, Uttar Pradesh 282001, India (rssneeraj@gmail.com).

a hash function. The encryption/decryption and hash algorithms employed are RSA and MD5, respectively, utilizing TCL, AWK, and C++ as programming languages. The simulation environment necessitates NS-2 version 3.34, a GCC compiler, and a C++/C editor, all running on a personal computer with the Red Hat Enterprise Linux (RHEL) operating system. The subsequent sections of the paper delve into a comprehensive review of related work, the cloning of wired networks, encryption/decryption algorithms, fundamental exchange mechanisms, and the proposed scheme. The modifications are then registered, and the mirroring/simulation environment is delineated. The paper concludes with an evaluation of the approach's performance and security aspects, culminating in a comprehensive wrap-up of the findings.

2. Related Work. The TCP/IP protocol stack or different layers of the OSI protocol have different protocols implemented in NS2. For a wired network, application layer protocols include telnet, FTP, and HTTP; transport layer protocols include UDP, TCP, and SCTP; IP layer protocols include Ping, IP, and so on; and data link layer protocols such as CSMA/CD, which were created by scientists and eventually included in NS2. Similarly, multiple protocols at various OSI or TCP/IP protocol stack layers have been suggested for wireless networks, including TORA, AODV, DSDV, DSR [19] and DYMO [14].

Subsequently, NS-2 integrated these procedures. Recent research endeavors have concentrated on devising new protocols for both wired and wireless networks [12], with some focusing on enhancing or reforming existing protocols. However, there remains a need for more extensive efforts to amalgamate security measures with data transmission in NS2. Scholars like JIANG Hong, YU Qing-song, and LU Hui from East China Normal University in Shanghai, China, have directed their research efforts in this direction. Their methodologies contribute to bolstering Ethernet security by introducing a group-based MAC critical selection procedure (GKSP) tailored for large Ethernet networks. They have implemented security measures at the data link layer, wherein security concerns are addressed hop-by-hop. In our security module, we uphold security at the IP layer, as it aligns with various application requirements, thereby ensuring end-to-end security oversight. The security module we've introduced aims to address data integrity and sender authentication concerns at the network layer, thus enhancing and integrating data security functionalities within NS-2. Leveraging freely available resources and adopting the same programming platform as NS-2, based on RHEL [10], has guided our approach. Creating a novel security module for NS2 represents a significant stride toward addressing the need for inherent security capabilities within the simulation tool. This module introduces crucial encryption, decryption, and sharing capabilities vital for ensuring secure communication across diverse applications. This paper endeavors to augment NS2's simulation prowess in modeling secure communication scenarios by prioritizing reinforcing security measures. In another study [17], researchers assess the performance of four MANET routing protocols under various CBR and TCP traffic patterns using NS2. While AODV outperforms CBR traffic, OLSR exhibits superiority in handling TCP traffic. OLSR showcases optimal performance in managing multimedia/TCP traffic, making it well-suited for internet traffic scenarios. In [4] the authors introduce a communication system for microgrids (MGs) using NS2. Each MG includes a central controller and multiple distributed generation units with local controllers. The system facilitates data exchange between these controllers and sensors/actuators, employing low-cost ZigBee devices. The system integrates duplicate acknowledgment and data management schemes to optimize data transfer despite ZigBee's limitations. It also ensures intelligent data routing in case of path or device failures. Performance metrics like Packet Delivery Ratio (PDR), throughput, and end-to-end communication time are evaluated to validate system effectiveness. The routing protocol selection in wireless networks, specifically on Mobile Adhoc Networks (MANETs) where throughput and minimal delay are critical [2]. Their study evaluates the performance of Adhoc On-demand Distance Vector (AODV) and Destination Sequenced Distance Vector (DSDV) protocols using an NS2 simulator, emphasizing parameters like packet throughput, jitter, and end-to-end delay. By designing a wireless network of mobile nodes with defined parameters, the researchers aim to provide a detailed comparative analysis to aid in selecting the most suitable protocol for optimal network performance.

The necessity for energy-efficient protocols in Wireless Sensor Networks (WSNs), focusing on the "Sensor-Medium Access Control" (S-MAC) protocol for its ability to reduce sensor node energy consumption is discussed in [9]. It highlights the challenge of preserving energy while meeting application demands in battery-constrained sensor nodes. The study introduces a novel energy-efficient clustering algorithm based on the LEACH protocol, comparing it with other clustering protocols such as LEACH-C, MTE, and Stats-Clustering through NS2

simulations. Additionally, the paper analyzes S-MAC using Network Simulator NS2.

NS2, developed by UC Berkeley, is an open-source simulator for Internet Protocol Networks. While it produces ASCII-formatted trace files capturing simulation events, analyzing these files can be challenging due to their textual format and large size. Q-Analyze, a trace file analysis tool, simplifies data extraction for performance metrics measurement and generates user-friendly simulation reports. Operating through three layers, Q-Analyze streamlines network performance study by providing intuitive GUI and focusing on algorithm development rather than data processing and metric calculation. NS2, developed by UC Berkeley, is an open-source simulator for Internet Protocol Networks. While it produces ASCII-formatted trace files capturing simulation events, analyzing these files can be challenging due to their sizeable textual format. Q-Analyze, a trace file analysis tool, simplifies data extraction for performance metrics measurement and generates user-friendly simulation reports. Operating through three layers, Q-Analyze streamlines network performance study by providing intuitive GUI and focusing on algorithm development rather than data processing and metric calculation [3].

3. Background.

3.1. Simulation of Wired Network. In NS-2, our objectives are achieved in two ways:

- The Otcl script has to be changed, but the built-in network modules in NS-2 do not need to be changed when they are sufficient to achieve the mirroring/simulation goal [11, 16, 20].
- If the pre-installed network modules/components are insufficient for mirroring/simulation, a new module/component must be made, or the current modules must be modified [21, 1, 13]. In other words, NS2 should be expanded by introducing a new Otcl class.

Adjustments must be made to the Otcl script to execute the simulation. Utilizing the existing components suffices for simulating the basic Mobile IPv4 protocol. However, for tailored functionalities such as specialized applications, business flows, agents, linkages, routing, and node models, thorough verification and compilation of these components are imperative to ensure seamless integration [5]. Post-simulation, analyzing trace files yields invaluable insights. Additionally, monitoring the network simulation process can be facilitated by utilizing NAM. The insights gleaned from simulation analysis are instrumental in discerning whether tweaks to the configuration topology and business simulation are warranted to initiate further simulations to achieve desired simulation outcomes.

3.2. Digest Generation using MD5 Algorithm. Professor Ronald L. Rivest from MIT is credited with creating MD5 [18]. MD5, an algorithm designed to generate a 128-bit fingerprint or message digest for any message, regardless of its length, is widely recognized for its computational in-feasibility in producing two messages with the same message digest or any message with a predetermined goal message digest. MD5 securely compresses large files in digital signature applications before encrypting them with a private key using a public-key cryptosystem such as RSA. Renowned for its reliability, MD5 surpasses checksum and many other widely used techniques in verifying data integrity. The core MD5 algorithm operates on a 128-bit state, segmented into four 32-bit words initialized to specific fixed constants. It iteratively processes each 512-bit message block, modifying the state. This processing occurs in four rounds, each comprising 16 similar operations: modular addition, left rotation, and a non-linear function known as F.

3.3. RSA Algorithm for Data Security. The RSA algorithm [18] was created by Ron Rivest, Adi Shamir, and Len Adleman, who created it in 1977. The RSA cryptosystem is the world's most used public key cryptography algorithm. It allows communication to be encrypted without exchanging a secret key individually. The RSA technique may apply to public key encryption and digital signatures. Its security relies on the difficulty of factoring huge numbers. Party A can deliver encrypted communication to Party B without exchanging secret keys. A encrypts the message with B's public key, and B decrypts it with only the private key he knows. RSA can also be used to sign a message, so A can sign a message using their private key, and B can verify it using A's public key.

Sender Side Encryption Process: Sender A adheres to these guidelines:

1. obtains the public key of recipient B i.e. (m, f).
2. a positive integer that symbolises the textual message pt, with $1 < pt < m$.

3. uses the formula to determine the cipher text d $d = pt^f \mod m$.
4. sends the encrypted text to B.

Receiver Side Decryption Process: Recipient B adheres to these guidelines:

1. uses the cipher text d and their private key (m, e) to calculate pt, using the formula $pt = d^e \mod m$.
2. extracts the plaintext from the representation of an integer (pt).

4. Proposed Work. To ensure message integrity and sender authentication, the self-defined security protocol Class, or *packet_sec*, implements hash and decrypt/encrypt functions. Furthermore, the security protocol keeps a sequence number seq for each node. If the sender delivers a packet, seq is increased by one and adds to the packet header information, allowing the receiving node to reorganize the packets. The general protocol believes a source file, *packet_sec.h/.cc* (the protocol's solution and realization) should be created. Algorithm 1 illustrates the definition of a security protocol in which the command and recv functions are inherited from the Agent class. The inter-layer communication mentioned is detailed within the Tcl code of NS2. The class definition referred to is available in *packetsec.h*. To enable this security protocol in Tcl, NS must recognize it as an Agent. The standard procedure to meet these criteria involves defining the packet sec class in C++ within Agent/packet sec. This facilitates the modification of the Tcl code and the definition of the security protocol. An inheritance relationship exists between Agent and packet sec, with packet sec inheriting from Agent.

Algorithm 1 Class `proto_sec_anal`: Public Agent

```
Class proto_sec_anal: Public Agent {
Public:
  void accept(Packet * k, Handler *)
  p: Packet accepted by the receiver
  Handler for processing
  int command(int argc, const char * const * argv)
  argc: Number of command-line arguments
  argv: Array of command-line argument strings
Private:
  uint32t serial_no
}
```

4.1. Implementation of the Digital Signature and Hash algorithms . Sender A computes the hash and acquires a digital signature as follows: In Algorithm 2, generating a hash on data retrieved from the TCL file and obtaining a digital signature for sender authentication is depicted. Initially, a packet is created using the authorized function allocpkt(). Subsequently, the packet header is accessed by creating an object named hdr with the type hdrpacketsec. The sending time is also provided in the send_time variable of the packet header, which functions as a timestamp. To obtain the hash value, the data, and its length are fed into the hashing algorithm, resulting in the hash value. This value is then utilized to verify the integrity of the message. Afterward, the RSA algorithm is applied to the data and passed to the decryption/encryption function, resulting in encrypted data. This encrypted data is assigned to the data variable in the packet header. Finally, these values are stored in packet header variables, and the packet is transmitted to the receiver [10].

Packet Verification by the Receiver. The process depicted in Algorithm 3 ensures that receiver B initiates a new hash generation and verifies the signature. Upon receiving the packet, the receiver initially stores the transmission time, sequence number, hash value, and encrypted contents in fresh variables. Subsequently, the receiver decrypts the data using the RSA method, forwarding it to a decryption function. If necessary, the function returns the original data. The decrypted data is then hashed using a hashing algorithm. Upon conclusion, the received hash value is compared with the newly computed hash value. If both values match, a signed acknowledgment is returned; otherwise, the message is adjusted.

The receiver sends an ack to relay the result. The approach presented in Algorithm 4 demonstrates how the receiver conveys the signature verification result. Initially, the receiver generates a new packet using the allocpkt() method and accesses the packet's header via the hdrret object. Later, ret variable is set to 1. So

Algorithm 2 Message Digest and Digital Signature Generation

```
if (strcmp(argv[1], "transmit") == 0) then
  // Make a one new packet
  Packet * packet = allocpkt()
  // Incorporate security packet header in the newly created packet
  packethdrsec * packet_hdr = packethdrsec::access(packet)
  // To let the receiving node know that it must send and acknowledge packets, set the 'bak' variable to 0.
  packet_hdr->bak = 0
  packet_hdr->sno = sno++
  // Put the time now in the "transmit_time" field.
  packet_hdr->transmit_time = Scheduler::instance().clock()
  // Move the copied info to the header.
  strcpy(packet_hdr->info, argv[2])
  //@@@@@@@@@@@@@@@ Hashing Operation @@@@@@@@@@@@@@@//
  packet_hdr->msgdigest = hashing (packet_hdr->info, (unsigned int) strlen (packet_hdr->info))
  //@@@@@@@@@@@@@@@ Securing the data @@@@@@@@@@@@@@@//
  data_encryption(packet_hdr->info)
  printf("Digital Signature Generated:")
  //@@@@@@@@@@@@@@@ Send a packet @@@@@@@@@@@@@@@@@//
  transmit(packet, 0)
  return (TCL OK)
end if
```

the receiver would not send another echo. Thereafter, report the transmitting time to the packet header's send_time variable. Finally, result is saved in the packet header's data field.

4.2. TCL File Modifications. TCL serves as a scripting language employed in developing network units and components. For our simulation, six nodes are established, four of which have security agent capabilities and are connected. Their connections are in the following order: node_0 to node_5 and node_1 to node_4. Later, invoke the function that sends the data. Lastly, delve into the recv function, which retrieves the variable's value and exhibits it on the Linux terminal upon successful execution. The Tcl file about digital signature verification and hash creation is depicted in Algorithm 5.

5. Security Module Registration. Marc Greis' example [18] illustrates introducing a new protocol to NS2. Initially, a new packet Class is created in the application folder ../apps. Following this, the packet name is inserted into the packet.h header file of NS2. Subsequently, modifications are made to the makefile to accommodate the creation of the new Class. Each newly created packet must be defined at the TCL layer by adding the default packet size value and name to the ns-default.tcl file. Finally, an entry for the newly created packet is added to the ns-packet.tcl file. Recompiling NS-2 will then prepare the new packet for simulation. A new packet type is devised for transmitting data.

6. Security and Permormance Analysis. After testing, analyze the security protocol's performance using experimental data statistics. Figure 6.1 shows the packet transmission status using the security protocol [19]. It may be seen in Figure 6.1. In security protocols, data is sent via unicast (Algorithm 6). When a node delivers a data packet, it unicasts it to directly linked nodes. If a node accepts a packet but has not yet transmitted it and the destination is not itself, it will forward it in unicast format. The procedure will continue until the data packet reaches its destination.

6.1. Environment Simulation. In the environment illustrated in Figure 6.1, NAM constructs a network with six nodes interconnected by a 5 Mbps full duplex link. Security agents are deployed on nodes n1, n4, and n5. Communication transpires between nodes n0 and n5 and nodes n1 and n4 via the link between n2 and n3. A drop-tail queue of 100 is employed between nodes n2 and n3. Following one minute of operation, a trace file is generated. Subsequently, the trace file undergoes processing using a text processing language like awk to derive the desired output.

Algorithm 3 Signature verification at destination

```
if packet_hdr->bak == 0 then
  double ttime = packet_hdr->transmit_time
  //@@@@@@@@@@ Packet data encryption/Decryption @@@@@@@@@@//
  char actual_info[128]
  char secure_info[128]
  strcpy(secure_info, packet_hdr->info)
  // Make a copy of the original packet's data
  strcpy(actual_info, packet_hdr->info)
  int accept_seq = packet_hdr->sno
  //@@@@@@@@@ Present the packet to the recipient node @@@@@@@@@//
  char var[105]
  unsigned int newdigest
  char verified_outcome[50]
  decryption(actual_info)
  newdigest = hashing(actual_info, strlen(actual_info))
  if newdigest == packet_hdr->msgdigest then
    printf("Signature matched")
    strcpy(verified_outcome, "Sender Authenticated")
  else
    printf("Sender did not sign the msg %dnn", newdigest)
    strcpy(verified_outcome, "Message Altered")
  end if
  // Drop the packet
  Packet::free(packet)
end if
```

Algorithm 4 Receiver sends ACK packet to confirm delivery of message

```
// Make a one new Packet
Packet * packetack = allocpkt()
// Access the new packet's header.
packethdrsec * packet_hdrack = packethdrsec::access(packetack)
packet_hdrack->bak = 1
// Enter the accurate value in the transmit time field
packet_hdrack->transmit_time = ttime
packet_hdrack->receive_time = Scheduler::instance().clock()
packet_hdrack->sno = accept_seq
strcpy(packet_hdrack->info, verified_outcome)
send(packet_hdrack, 0)
```

6.2. Analysis of Security Protocol. Figure 6.1 depicts the transmission of a unicast data packet. After executing the security procedure, Algorithm 6 illustrates concise results. Initially, node n0 transmits the data to node n5 utilizing the hash and RSA technique. Node n5 then receives the packet and decrypts the message using the RSA technique, retrieving the original contents. Subsequently, a hash of the decrypted data is generated and compared to the received hash value. If a match is found, the node acknowledges receipt of a signature-validated message; otherwise, a message indicating data alteration is transmitted. Similarly, node n1 communicates with node n4. This process iterates in reverse order accordingly. The time required for encryption, decoding, and hash calculation was also computed.

7. Conclusion. In summary, this study delves into the simulation intricacies of a wired network, incorporating two pivotal protocols: message integrity and sender authentication. We meticulously expound upon the simulation methodology within NS2, elucidating our analytical approach to interpreting the results. The visualization of the simulation process is facilitated through NAM, while Awk emerges as the tool of choice for

Algorithm 5 Receiver sends ACK packet to confirm delivery of message

```
set q1 [new Agent/pkt_sec]
$ns attach-agent $m1 $q1
$q1 set class 2
set q2[new Agent/pkt_sec]
$ns attach_agent $m2 $q2
$q2 set class 2
set q3[new Agent/pkt_sec]
$ns attach-agent $m5 $q3
$q3 set class 3
set q4[new Agent/pkt_sec]
$ns attach-agent $m6 $q4
$q4 set class 3
// Link the two agents together
$ns connect $p0 $p3
$ns connect $p1 $p2
// Plan your events
for {set j 1}{$j<2} {incr j}{
set outcome [expr $j/2]
$ns at [expr $outcome + 0.04] "$q1 transmit thisispk"
$ns at [expr $outcome + 0.40] "$q2 transmit thisispks"
$ns at [expr $outcome + 0.80] "$q3 transmit thisis"
$ns at [expr $outcome + 1.20] "$q4 transmit thisis..."}
// function 'accept1' for the 'Agent/pkt_sec' class
Agent/pkt_sec instproc accept1 (from rtt) {
$self instvar node
puts "node [$node id] obtained a secret Key from
$from with trip-time $rtt ms"}
// function 'accept' for the 'Agent/Packet_sec' class
Agent/pkt_sec instproc accept (from rtt
mess originmess hash) {
$self instvar node
puts "node[$node id]collected_packet from
$from with trip-time $rtt ms - contend: $mess -
decrypted $originmess -hash: $hash"}
```

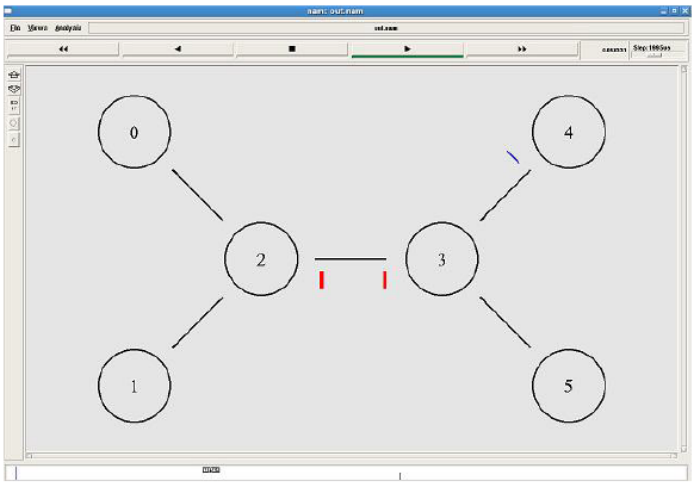

Fig. 6.1: Creating an environment using Network Animator Tool (NAM)

Algorithm 6 Outcome of security protocol after execution

```
Message transmit: "Thisispk" along with message digest using MD5 algorithm
Message Digest: 906d9a4206f569725ffb4b2b178baba6 & SIGNATURE CREATED
Message Digest Collected by Receiver: 906d9a4206f569725ffb4b2b178baba6
SIGNATURE IS VERIFIED
Node 5 obtained a packet from 0 with a 31.1 ms travel time. -
contend encrypted_data: sCcbIEYv8pe/9+XNCpFrTMW0Y -
decrypted: Thisispk
node 5 sent an ack packet to node 0 with a travel time of 62.2 ms. -
contend: SENDER_AUTHENTICATED
Message transmit "Thispks" along with message digest using MD5 algorithm
Message Digest: 453558b732457b31e8eb083afdf5435f and DIGITAL SIGNATURE CREATED
MD5 Digest at Receiver Side: 453558b732457b31e8eb083afdf5435f
SIGNATURE VERIFIED
With a journey time of 31.1 ms, node 4 received a packet from 1. -
contend encrypted_data: pMECBU04/RYvM94XwLrpzlsLPj1J6ORNxtsu1ExN
decrypted: Thisispks
Node 1 obtained a packet from 4 with a 62.2 ms travel time. -
contend: SENDER_AUTHENTICATED
```

trace file post-processing and nuanced comparisons. Expanding our investigation, we systematically manipulate network topologies and data flows in diverse configurations, consistently yielding closely aligned results. The successful integration of a security module into NS2 marks a significant milestone, enabling its versatile application across a spectrum of data security-intensive scenarios. This juncture affords us the capability to scrutinize and assess various applications post-implementation, delving into facets such as security overheads, packet loss, required bandwidth, throughput, and other pertinent metrics. Within the implemented module, the MD5 algorithm serves to generate a 128-bit digest, while the RSA algorithm assumes the role of our encryption mechanism. It is imperative to note that the flexibility exists to substitute the RSA algorithm with any public key algorithm; the current preference for RSA is rooted in its pragmatic simplicity. Emphasizing its specificity to wired networks, our future endeavors entail extending this module's support to encompass wireless networks.

Anticipating the future, our research trajectory extends beyond the current scope, entering a phase of advanced exploration and innovation in the realm of network security simulations. With a particular emphasis on the dynamic difficulties and developing threats in the cybersecurity landscape, we are dedicated to an unyielding pursuit of thorough studies that delve deeper into the complex details of applications post-implementation. Our commitment to excellence is reflected in the ongoing refinement and enhancement of our simulation methodology and analytical frameworks. This involves continuous integration of cutting-edge technologies, adaptive algorithms, and advanced cryptographic techniques to fortify the security module. Through a continuous feedback loop of simulation results and empirical insights, our aim is to fine-tune our models, ensuring they remain resilient in the face of emerging security threats. Furthermore, our future endeavors include the exploration of novel encryption algorithms and cryptographic primitives, seeking to elevate the security standards within network simulations. We anticipate delving into the realm of quantum-resistant cryptography, considering the evolving landscape of quantum computing and its potential implications for network security. In our quest for an evolved security paradigm, we are committed to extending the application of our module to diverse network architectures, including wireless networks. This expansion will involve adapting the existing security framework to the unique challenges posed by wireless communication, such as channel vulnerabilities and mobility issues. As proceed, collaborative efforts with industry partners, academic institutions, and cybersecurity experts will play a pivotal role. This collaborative approach will not only validate the robustness of our simulations but also foster a knowledge-sharing ecosystem, contributing to the collective advancement of network security research.

In conclusion, our research trajectory is characterized by a forward-looking perspective, driven by an unyielding commitment to advancing the field of network security simulations. Through continuous refinement, exploration of emerging technologies, and collaborative partnerships, we aspire to shape a future where network simulations serve as a cornerstone for developing resilient and adaptive cybersecurity solutions.

REFERENCES

[1] D. Adami, C. Callegari, D. Ceccarelli, S. Giordano, and M. Pagano, *Design, development and validation of an ns2 module for dynamic lsp rerouting*, in Computer-Aided Modeling, Analysis and Design of Communication Links and Networks, 2006 11th International Workshop on, 2006, pp. 72–77.

[2] N. Aggarwal, T. S. Chohan, K. Singh, R. Vohra, and S. Bahel, *Relative analysis of aodv & dsdv routing protocols for manet based on ns2*, in 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), IEEE, 2016, pp. 3500–3503.

[3] Q. Al-Shidi, A. Alburaiki, H. Shaker, and B. Kumar, *Q-analyze tool to detect malicious and black hole nodes in ns2 simulation for aodv*, in 2018 International Conference on System Modeling & Advancement in Research Trends (SMART), IEEE, 2018, pp. 140–146.

[4] P. O. Bhalerao, *Communication system design and simulation for future micro grids in ns2*, in 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), IEEE, 2016, pp. 688–690.

[5] N. Glance, D. Snowdon, and J.-L. Meunier, *Pollen: using people as a communication medium*, Computer Networks, 35 (2001), pp. 429–442.

[6] M. Greis, *Tutorial for the network simulator 'ns'*. `http://www.isi.edu/nsnam/ns/tutorial/`, 2006.

[7] Z. Ishak, N. Din, and M. Jamaludin, *Ipqit: An internet simulation kit based on ns2*, in Telecommunications and Malaysia International Conference on Communications, 2007. ICT-MICC 2007., 2007, pp. 489–493.

[8] H. Jiang, Q.-s. Yu, and H. Lu, *Simulation and analysis of mac security based on ns2*, in Multimedia Information Networking and Security, 2009. MINES '09., vol. 2, 2009, pp. 502–505.

[9] K. H. Krishna, T. Kumar, and Y. S. Babu, *Energy effectiveness practices in wsn over simulation and analysis of s-mac and leach using the network simulator ns2*, in 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), IEEE, 2017, pp. 914–920.

[10] P. Kumar, S. Maheshwari, and H. K. Dubey, *Development and validation on ns2 protocol for data security at network layer*, in International Conference on Computing, Communication & Automation, 2015, pp. 529–534.

[11] S. Kumar, R. Rathy, and D. Pandey, *Traffic pattern based performance comparison of two reactive routing protocols for ad hoc networks using ns2*, in Computer Science and Information Technology, 2009. ICCSIT 2009., 2009, pp. 369–373.

[12] G. Li and J. Chen, *The research of routing algorithms based on ns2 in mobile ad hoc networks*, in Software Engineering and Service Science (ICSESS), 2011., 2011, pp. 826–829.

[13] M. Qadeer, V. Sharma, A. Agarwal, and S. Husain, *Differentiated services with multiple random early detection algorithm using ns2 simulator*, in Computer Science and Information Technology, 2009. ICCSIT 2009., 2009, pp. 144–148.

[14] M. Q.-x. Qu and L. Xu, *Dymo routing protocol research and simulation based on ns2*, in Computer Application and System Modeling (ICCASM), 2010., vol. 14, 2010, pp. V14–41 – V14–44.

[15] Z. Qun and J. Wang, *Application of ns2 in education of computer networks*, in Advanced Computer Theory and Engineering, 2008. ICACTE '08., 2008, pp. 368–372.

[16] C. P. Reddy and C. P. Reddy, *Performance analysis of adhoc network routing protocols*, in Ad Hoc and Ubiquitous Computing, 2006. ISAUHC '06., 2006, pp. 186–188.

[17] S. J. Soni and J. S. Shah, *Evaluating performance of olsr routing protocol for multimedia traffic in manet using ns2*, in 2015 Fifth International Conference on Communication Systems and Network Technologies, IEEE, 2015, pp. 225–229.

[18] W. Stallings, *Cryptography and Network Security*, fourth edition ed., 2006.

[19] A. Tuteja, R. Gujral, and S. Thalia, *Comparative performance analysis of dsdv, aodv and dsr routing protocols in manet using ns2*, in Advances in Computer Engineering (ACE), 2010., 2010, pp. 330–333.

[20] S. Xu and Y. Yang, *Protocols simulation and performance analysis in wireless network based on ns2*, in Multimedia Technology (ICMT), 2011., 2011, pp. 638–641.

[21] S. Zhao, P. Wang, and J. He, *Simulation analysis of congestion control in wsn based on aqm*, in Mechatronic Science, Electric Engineering and Computer (MEC), 2011., 2011, pp. 197–200.

Edited by: Kavita Sharma
Special issue on: Recent Advance Secure Solutions for Network in Scalable Computing
Received: Apr 6, 2024
Accepted: Jun 12, 2024