ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 10, December 2012

A Congestion Control And Policy Enforcing Mechanism For Reliable Routing Using SATEM In Mobile Ad Hoc Networks

M. Rajalakshmi, M. Manikandan

Abstract— **A Policy enforcing mechanism in MANET is performed to provide the secure communication. If two node enforce the same set of policies for two instance of application and also if the underlying protocol used by the application are the same then these two instance running on different nodes can engage in communication. Nodes can be a member of multi-tier network at the same time. A policy enforcing in MANETs challenging because lack of they infrastructure so introduce a Satem. Satem verifies the trustworthiness of the new node when it joins the tier. Policy enforcing mechanism based on Satem, a kernel-level trusted execution monitor. The security provided by the mechanism is validated by performing the security analysis. Performance of the network is evaluated using network simulator.**

Index Terms— **Satem, MANETs, Policy Enforcing Mechanism, Routing, Congestion control**

I. Introduction

Mobile Ad-hoc networks (MANETs) are infrastructure-free wireless communication networks. MANETs are considered as ideal technology for instant communication networks in both military and civilian applications. Now a days, tactical military networks are the main application area of MANETs. Tactical military networks, having critical operation environments, require very high security and performance together. A typical text on Mobile Ad hoc NETworks (MANETS) will describe such networks as simply being "a collection of mobile nodes, communicating among themselves over wireless links. It provides a number of benefits, which make it suitable for MANETs. First, policy enforcement in the multi-tier networks is entirely distributed without relying on any central trusted choke points (e.g., firewalls, proxy). Second, the trusted networks are self-organized. Third, the multi-tier trust enables flexible enforcement of complex policies, which can be defined across various interdependent protocols and enforced independently, node by node. Moreover, nodes running multiple applications can join multiple trusted networks, each enforcing policies for different applications without interfering with each other applications.

Satem, a **S**ervice-**a**ware **t**rusted **e**xecution **m**onitor that guarantees the trustworthiness of the service code across a whole transaction. The Satem architecture consists of an execution monitor residing in the operating system kernel on the service provider platform, a trust evaluator on the client platform, and a service commitment protocol.

The monitor resides in the OS kernel of the service provider and has two main goals: (1) provide a guarantee to the service requester that only trusted code will be executed by the service during the transaction, and (2) enforce fail-stop protection of the service during transactions.

Congestion control is associated to controlling traffic incoming into a telecommunication network. To avoid congestive collapse or link capabilities of the intermediate nodes and networks and to reduce the rate of sending packets congestion control is used extensively.

The policy certificate is a structured set of information that conveys the policy assigned to an entity. A set of policy entries comprises the policy. Besides the policy entries, the elements of the certificate indicate from the issuer and to the owner the certificate was issued, its period of validity, and supplemental information needed to establish its authenticity and apply the policy.

A digital signature over the other elements protects the certificate from tampering as well as attempts to forge the issuer's signature on another certificate. In order to verify the signature and establish the authenticity of the certificate, a device must hold the corresponding public key of the issuer.

The policy enforcement mechanism resides on the handheld device and ensures that the user adheres to the security administrator's security policy settings specified within a valid policy certificate stored on the device. The mechanism starts up as the device is initialized and checks the issuer's signature on the policy certificate for authenticity, the well-formedness of the contents, and whether the validation period is in effect. If a policy certificate is not held or is found to be invalid, the enforcement mechanism applies a default policy having limited privileges.

To ensure trusted policy enforcement, we augment each node with a trusted agent, which protects the policy enforcement components from being compromised. When a node joins a trusted tier, its trusted agent helps establish trust by proving the execution of a correct trusted agent, a trustworthy policy enforcing software component, and the right policy. Moreover, it ensures that the integrity of the agent, the enforcer, and the policy will not be compromised. This is possible because the trusted agent is part of the operating system kernel and guarantees the integrity of the kernel and all programs involved in policy enforcement.

Therefore, it can foil attacks, including those launched by local users, to tamper with the enforcer or the policy being enforced. If any of these components is compromised, the trusted agent will disconnect the node from the trusted network.

The trusted agent is built on top of Satem , our trusted execution monitor based on a low-end trusted hardware, Trusted Platform Module (TPM) specified by the Trusted Computing Group (TCG) . Due to its low cost and broad support by computer makers, the TCG TPM has been already integrated in many laptops.

II. PROPOSED SYSTEM

The focus of the trusted agent is to provide a fail-stop protection mechanism to enforce commitments. Our implementation is integrated into the OS kernel in many places by inserting checkpoints to kernel calls system performance by adding latency to 1) the kernel call execution due to enforcing the Satem commitment; 2) joining the trusted network due to verifying trustworthiness during JOIN and MERGE protocols; and 3) the network communication due to enforcing the policy and computing and verifying the MAC for application messages. the cost of network creation in terms of both successful ratio of JOIN and MERGE operations and the latency it takes for a node to join the network.

The completion ratio of JOIN (or MERGE) is defined as the total number of nodes that successfully join the network against the number of nodes that apply to join (or merge into) the network. To test MERGE, we first set all nodes in the network to be the members of the old tier. We randomly selected one node and updated its membership to become the first node of the new tier. Then, this node automatically started the MERGE process with other nodes.

In the future, we plan to implement a stand-alone enforcer as a transparent application proxy. In this way, the application request is redirected to its local system, which communicates with the application on the remote node. One way to achieve this is to establish the mapping between the application and its enforcer when the enforcer registers with the tier manager. the source node monitor the data loss count in each intermediate routing nodes in the network for a periodic interval. We set the threshold limit for the packet loss. If the Data loss get exceeded in the network the source find that buffer overflow attacker in the routing path. In the path finding the alternate secured path is find out in the network by eliminating the buffer overflow attacker in the routing path. The attacker's id is stored in the attacker file along with the detected node and the time of identification.

A. System Model

Satem is to achieve trusted services code execution across client service transaction. It comprises of components on both the service provider and the service requester. The service provider components include TPM (Trusted Platform Module) a trusted execution monitor and a commitment for each protected services, on the service requester includes a trust evaluator and a trust policy Fig.1shows on example of system model

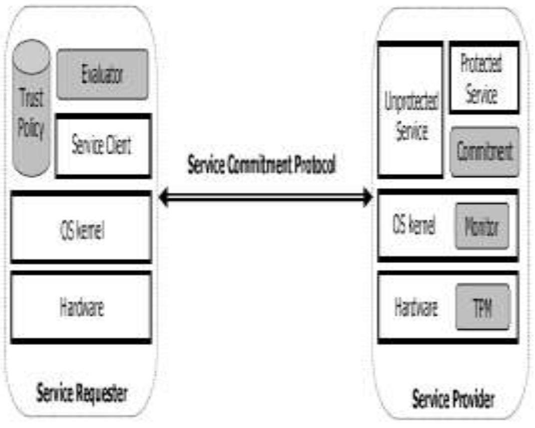

Fig.1 System Model

The TPM helps to establish trust on the os kernel and the monitor by attesting each component, it is assume that the attestation it conducts and the report it produces cannot be tempered. Trust Execution Monitor resides in the os kernel of the service provider and has two main goals i) provide a guarantee to the service request that only trusted code will be executed ii)enforce fail stop protection of the service during transactions.

Each protected service has an associated commitment that describes all the code, the service may execute in its entire lifetime. The evaluator verifies the TPM certificates, attestation report, and service commitment and also checks the local trust policy if everything is correct the user is convinced that the service will only execute trusted code.

III. Satem

Satem is a Service aware trusted execution monitor that guarantees trust worthiness of a service code across a client and server transaction. Some policies are used in Satem authentication, integrity, non-reputation. Authentication is the act of claimed identity , integrity means the assurance that data received a exactly send and non-reputation means protection against denial by sender and receiver.

A. Application Execution

The ad hoc network is in operation, numbers of applications like email, instant messaging, ftp and many others have to be started by the nodes in the network.

All of the participating nodes are ad hoc in nature so it is advisable to ensure the validity of the target node before staring any type of application execution as an interaction with the target node.

B. Routing Environment

The ad hoc network is in operation, there is a lot of packet flows over the network. The packet follows the path as per the routing protocol from node to node. In this context before forwarding the packet the source first gets the trust value on the receiver and is allowed to forward only if the trust value is above the threshold specified as per the policy. As the trust value is the result of past interactions so any misbehaving node can be excluded by this validation on the basis of trust.

C. Authentication

To accept or reject a public key certificate depends on the rust value of introducing node. Therefore the nodes involved in decision making is the value of trust that node s has on the originator.

D. Pick the Best

It is possibility the nodes have number of options i.e. number of nodes in the, for an interaction or getting a service from it. In order to select among them, one of the criteria is to go ahead with the node for which the initiator has the highest trust value. So it leads to choosing the best among the available choices.

Satem is used to two protocols, 1)join, 2)merge.

E. Joining a Tier

The JOIN protocol is used when a node wants to join a trusted tier for the first time. The new node communicates with a member node of the trusted tier. The member node has to verify that the new node is trustworthy to enforce the tier policy. At the same time the new node must also verify the trustworthiness of the member node. we assume that Node 2 is already a member of a trusted tier and Node 1 wants to join this trusted tier. In this example, we also assume that each application comes with an associated policy, which is stored on each node together with the application.

F. Merging a Tiers

Two tiers enforcing the same policy but using different keys can be united by the MERGE protocol. Every node has an equal opportunity to establish a trusted tier. To prevent multiple nodes from establishing the same trusted tier at the same time, the originator may first query its neighborhood for the existing tier. However, this method does not work if two nodes are not reachable from each other. As a result, they will create two trusted tiers running the same application and enforcing the same policy, but holding different trusted tier keys.

IV. SATEM POLICES

A. Policy Certificate

The policy certificate is a structured set of information that conveys the policy assigned to an entity. A set of policy entries comprises the policy. Besides the policy entries, the elements of the certificate indicate from whom (i.e., the issuer) and to whom (i.e., the owner) the certificate was issued, its period of validity, and supplemental information needed to establish its authenticity and apply the policy.

A digital signature over the other elements protects the certificate from tampering as well as attempts to forge the issuer's signature on another certificate. In order to verify the signature and establish the authenticity of the certificate, a device must hold the corresponding public key of the issuer.

B. Policy Specification Language

The policy language follows a grant-style form of specification by which security relevant actions are denied on a device unless enabled by a policy entry. Policy entries are a triple of action, source, and target fields Source refers to objects on the device, such as a calendar or address book application on a PDA, able to perform some action. Action refers to security-relevant operations performed with the device, such as synchronizing with a home platform, or beaming information. Target is an optional field that refers to external points of interface or reference needed to complete the semantics of the operation, typically by adding more specificity such as qualifying the origin or destination of data to be exchanged.

C. Policy Distribution

The policy rules ultimately affect the behavior of a device, they must always be protected from tampering and forgery. Using a policy certificate to convey policy entries provides inherent protection, requiring only an authenticated distribution from a trusted source. This facilitates the distribution process greatly, allowing a variety of ways for policy certificates to be handled and distributed, including authenticated Web services, secure electronic mail, secure file transfer, and authenticated device synchronization. Synchronization is the term used to describe the common means of coordinating the update of the information content on a host and handheld device to the same level. To move the policy certificate from a user's desktop, a centralized policy server, or some other host onto a device, we devised a special policy conduit for the synchronization process.

D. Policy Enforcement

The policy enforcement mechanism resides on the handheld device and ensures that the user adheres to the security administrator's security policy settings specified within a valid policy certificate stored on the device. The mechanism starts up as the device is initialized and checks the issuer's signature on the policy certificate for authenticity, the well-formedness of the contents, and whether the validation period is in effect.

If a policy certificate is not held or is found to be invalid, the enforcement mechanism applies a default policy having limited privileges. As mentioned earlier, the default policy blocks all information flows to external interfaces with the exception of allowing limited synchronization to obtain a valid certificate. When parsing the policy certificate, the enforcement mechanism extracts and sets aside the policy entries in a table for later use. It also provides a display for the user to review the entire certificate, including its policy contents.

V. IMPLEMENTATION

It consists of the Satem based trusted agent and the tier manager. To evaluate the performance, we also implemented enforcers for two MANET applications: DSDV for ad hoc routing and P2P file sharing. The Destination-Sequenced Distance-Vector (DSDV) Routing Algorithm is based on the idea of the classical Bellman-Ford Routing Algorithm. Obtain a table that contains shortest path from this node to every node Incorporate table updates with increasing sequence number tags Prevent loops, Counter the count-to-infinity problem, Faster convergence, Exchange table between neighbors at regular time interval., Two types f table updates are Incremental update and Full dumps update.

Incremental updates are takes a single network data packet unit (NDPU) When no significant change in the local topology and Full dumps updates are Takes multiple NDPUs: When local topology changes significantly Or incremental updates require more than a NDPU. Table updates are initiated by the destination with the new sequence number which is always greater than the previous one Single link break cause propagation of table update information to the whole networkThe changed node informs neighbors about new shortest path while receiving the table update message. Some advantages of dsdv routing protocol route setup process is very fast Make the existing wired network protocol apply to ad hoc network with some modifications.

The completion ratio of JOIN (or MERGE) is defined as the total number of nodes that successfully join (or merge into) the network against the number of nodes that apply to join (or merge into) the network. To test MERGE, we first set all nodes in the network to be the members of the old tier. We randomly selected one node and updated its membership to become the first node of the new tier. Then, this node automatically started the MERGE process with other nodes.

VI. SIMULATIONS

In simulations, there are a variety of software applications or platforms widely available, such as NS2, QualNet, OMNeT++, we decided to use NS2 simulator. The cost of cryptographic operations associated with JOIN, MERGE, and enforcement cannot be ignored but NS-2 does not account for execution time, we add certain latency for these operations.

The mean latencies are: 1150ms for JOIN, 180ms for MERGE, and 0.15ms for enforcement. The completion ratio of JOIN (or MERGE) is defined as the total number of nodes that successfully join (or merge into) the network against the number of nodes that apply to join (or merge into) the network. To test MERGE, we first set all nodes in the network to be the members of the old tier. We randomly selected one node and updated its membership to become the first node of the new tier. Then, this node automatically started the MERGE process with other nodes. The latency for a node to join the network is measured as follows:

where we denote t_i as the time node i joins the network, t0 as the time the originator initiates the network, and $dist_i$ as the number of steps the join invitation message has traversed before reaching node i. In another words, we measured the latency per hop. The reason to do so is because obviously the more number of hops a node is away from the network originator, the longer it takes for it to join the tier. We do not count nodes that fail to join the network since the latency for these nodes is infinite.

In the simulations results first form the node in the network, after that find the originator. Originator fine the which node is trust and untrust agent node and which routing protocol are used. Select the source node and sends the packet to corresponding destination node.
Each node maintain the buffer and set some threshold value in the node, the source node monitor the data loss count in each intermediate routing nodes in the network for a periodic interval. We set the threshold limit for the packet loss. If the Data loss get exceeded in the network the source find that buffer overflow attacker in the routing path. The alternate secured path is find out in the network by eliminating the buffer overflow attacker in the routing path.

VII. LIMITATIONS

Satem only ensures that a protected service cannot load untrusted code from the disk. It is unable to tackle attacks, like buffer overflow, that can cause the protected service to run arbitrary code without changing its disk image. Satem only mitigates the problem in two aspects. First, Satem may reveal the code that has known buffer overflow vulnerabilities by attesting it to the user. Hence, the user can avoid trusting the vulnerable code. Second, in the case of a successful buffer overflow attack, the attacker runs her own code on the service stack without being caught by Satem. But due to the limited size of the stack, the attacker's code typically has to call other local programs on the service provider to make the attack meaningful. Satem restricts the attacker's capability of launching arbitrary local code.

Satem-based method to implement network access control in ad hoc networks. This paper further extends the idea in two fronts. First, the nodes can now verify the trustworthiness of each other at any layer and for any application rather than just at link layer. This enhancement enables finer grained control of network establishment. Second, the policies can be associated with any applications rather than just with the network layer. This makes it possible to regulate the communication in any application protocol.

The tier keys are protected in memory. However, a recent study demonstrates the possibility of retrieving the keys directly from DRAM, since DRAM still retains the content even after being pulled out from the motherboard. Fully addressing this vulnerability may require architectural changes to DRAM to make it lose memory faster. Satem kernel code is not modularized due to the need of inserting integrity check points at various places in the kernel. This makes the code difficult to port and modify. We are exploring other methods such as Linux Security Module for improvement. In the current prototype, we implemented the enforcer by hard coding the policy enforcing function in the application source code. This is inflexible since changing the policy may require modifying the application. In the future, we plan to implement a standalone enforcer as the transparent application proxy.

VIII. Conclusion

Satem, a novel service-aware trusted execution monitor that guarantees trusted service code execution across client-service transactions. Users establish trust with the services through a service commitment protocol executed before starting any new transaction.

Satem only ensures that a protected service cannot load untrusted code from the disk. It is unable to tackle attacks, like buffer overflow and block hole attack, that can cause the protected service to run arbitrary code without changing its disk image.

The block hole attack is a severe attack that can be easily employed against routing in mobile ad hoc networks. A black hole is a malicious node that falsely replies for any route requests without having active route to specified destination and drops all the receiving packets. If these malicious nodes work together as a group then the damage will be very serious. This type of attack is called cooperative black hole attack.

Acknowledgment

Our thanks to the experts who have contributed for the development of a policy enforcing mechanism for the reliable routing using SATEM and its simulated solution.

REFERENCES

[1] G. Xu, C. Borcea, and L. Iftode, “Satem: A Service-Aware Attestation Method toward Trusted Service Transaction,” Proc. IEEE Symp. Reliable Distributed Systems (SRDS), pp. 321-336, Oct.2006.

[2] Wayne A. Jansen, Tom Karygiannis Assigning and Enforcing Security Policies on Handheld Devices

[3] S.L. Keoh, E. Lupu, and M. Sloman, “Peace: A Policy-Based Establishment of Ad-Hoc Communities,” Proc. 20th Ann. Computer Security Applications Conf. (ACSAC), pp. 386-395, Sept. 2004.

[4] Vishnu Kumar Sharma and Dr. Sarita Singh Bhadauria, Mobile Agent Based Congestion Control Using Aodv Routing Protocol Technique For Mobile Ad-Hoc Network

[5] Hesiri Weerasinghe and Huirong Fu, Preventing Cooperative Black Hole Attacks in Mobile Ad Hoc Networks: Simulation Implementation and Evaluation.

[6] Charles E. Perkins, Pravin Bhagwat, Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) For Mobile Computers

[7] Yan Sun, Wei Yu, Zhu Han, and K. J. Ray Liu Trust Modeling and Evaluation in Ad Hoc Networks.

[8] Yong Lee, Goo Yeon Lee, Design and Performance Evaluation of a Scalable Authentication Protocol in Mobile IP.

[9] Amandeep Verma1 and Manpreet Singh Gujral Trust Oriented Security Framework For Ad Hoc Network.

[10] Capkun.S, Buttyan.L, and Hubaux J.P, “Self-Organized Public- Key Management for Mobile Ad Hoc Networks,” IEEE Trans.Mobile Computing, vol. 2, no. 1, pp. 52-64, Jan.- Mar. 2003.

M. Rajalakshmi received her B.E degree in Computer Science Engineering from Anna University Chennai in 2011 and i s currently doing her M.E degree in Adhiyamaan College of Engineering.

M. Manikandan received his B.E degree in Computer Science Engineering from Anna University Chennai in 2005 and M.E degree in in Computer Science Engineering from Anna University Chennai in 2007 and currently working as Assistant Professor n Adhiyamaan College of Engineering.