Design and Implementation of Application-based Secure VLAN

Minli Zhu
Computer Science Department
University of California, Riverside
mzhu@cs.ucr.edu

Mart Molle
Computer Science Department
University of California, Riverside
mart@cs.ucr.edu

Bala Brahmam
TATA Consultancy Services
balad@tcs-america.com

***Abstract*— Normally, security and access control policies rely on features from the host operating system. But what if the owner of the host is untrustworthy, or the host is simply too "dumb" to support such policies? In this paper, we investigate the feasibility of using the network to enforce these policies. First, we describe an application-based secure VLAN architecture that can be used to completely secure a particular network service from unauthorized access, without making any changes to the hosts and requiring only minimal changes to existing standardized LAN components. Second, we present a case study to show how our approach can be applied to IP Telephony. Finally, we describe our experiences from a prototype implementation of the S-VLAN concept.**

Keywords: Service-based VLAN overlay; tagged VLAN; Link Layer Network Address Translation (NAT); Link Layer Security; tunneling; proof-of-concept test bed

I. INTRODUCTION

In any computing environment that supports multiple processes, there must be some mechanism for *resource allocation* among competing processes, as well as the enforcement of *access control policies* based on the priviledges assigned to that process compared to the requirements of the resource. In a traditional timesharing operating system, these policies are enforced by the kernel, e.g., when the application program makes a system call. In the networking context, firewalls play a similar role in policy enforcement for sessions and/or individual datagrams attempting to cross the boundary separating the "inside" and "outside" worlds.

At the same time, the individual hosts often view the network as an untrustworthy resource. Once again, the host operating system has the primary responsibility for managing network resources, such as the configuration of the network interface to detect and block unauthorized remote accesses and also protecting the integrity of its own network traffic via encryption (e.g., IPSec, SSL, VPN's etc.) or other means.

In this paper, however, we consider a somewhat different view of the environment. What if we cannot rely on the host operating systems to enforce the desired resource allocation and access control policies, but we do have complete control over (and can trust) the switches at the edge of the network? What policy classes can be implemented in the networking hardware, without making any changes to (or trusting) the hosts?

To motivate this work we provide two possible application domains. First, we consider the problem of attaching *limited functionality embedded computers* to the network. For example, a basic IP telephone could be a single-purpose computer packaged in a telephone-shaped "sealed box" that you simply plug into a powered-Ethernet network port. End users are not capable of installing additional software (e.g., encryption tools) beyond the basic vendor-supplied system. Moreover, such devices — along with a variety of even-simpler networked controllers, such as the canonical networked refrigerator or light switch — are intended for use on a "safe" private intranet, and hence are not "hardened" to defend themselves against a determined external attacker.

Rather than assuming that the individual hosts are weak and in need of protection by the network, our second application considers the case where the individual hosts are simply not trusted by network administrator. For example, *visitor networks* (or hot-spot networks) [23] provide generic Internet access in a variety of public places, including university libraries, airports and hotels. Unlike a private enterprise network, where a boundary firewall can be effectively used to protect most private network services from Internet intruders, we cannot be sure that a visitor's computer is properly configured, that proper secu-

rity mechanisms are enabled, and that the visitor does not have malicious intent to launch a Denial-of-Service attack or attempt to monitor and/or interfere with communications among other users. Moreover, even in private enterprise networks it is possible for a host to become infected by a virus or worm, or for the security of that host to be breached by an outside attacker who has taken advantage of known flaws in an obsolete and/or poorly administered system. In either case, since the number of edge switches in a typical network is much smaller than the total number of hosts connected to those switches, there is a potential for significantly reducing the amount of effort required to maintain resource allocation and access control policies if it could be shifted to the level of the network switches rather than depending on the proper operation of every individual host.

The rest of the paper is organized as follows. In Section II we provide some background to the problem, and compare our approach to existing work. In Section II-B we state our research problem and give a overview of our approach. In Section IV, we describe the design principles for our secure VLAN tunnels, which are the key component in our approach. In Section V, we discuss our implementation considerations. In Section V, we show the experiment results from our proof-of-concept prototype test bed. and make the discussion according to experiment results. Finally in Section VII, we make a conclusion.

II. Background and Related Work

Network security mechanisms can be classified as either *passive* mechanisms, including accounting and intrusion detection; or *active* mechanisms, i.e., prevention mechanisms. Among prevention mechanisms, we can further divide them into security mechanisms that can be provided entirely by network infrastructure, and those that require the participation of the end hosts. In this paper, we are interested in *intrastructure-based active mechanisms*, for reasons we explained in section I.

A. Infrastructure-Based Security Mechanisms

Boundary firewalls are the most popular mechanisms adapted by private enterprise networks, as they can effectively filter malicious traffic from Internet. Proxy firewalls can even utilize network address translation (NAT) to masquerade the hosts' real IP address and port number. A firewall cannot prevent attacks or security holes that originate from inside the network. However, distributed firewalls [5] can be employed to control traffic traveling in and out of a single host connected to the network.

Internet Protocol Security Architecture (IPSec) [3] is a point-to-point protocol, which represents the current best practice for providing Network Layer Security between hosts. In addition, IPSec has a *tunnel mode*, which can be used by VPN gateways to create a secure virtual link across the Internet between separated sub networks belonging to the same enterprise. In this case, hosts inside each sub network need no changes. Unfortunately, IPSec is quite complex, so it is hard to implement and configure correctly. Moreover, IPSec expands each packet by adding a large AH or ESP header ($\geq$20 bytes). This expansion adds considerable overhead, since most Internet packets are short[7] — especially for real-time applications such as VoIP packets. Conversely, if the application chooses a large packet size for efficiency, then the extra header size for IPSec may lead to packet fragmentation. Since IPSec is point-to-point protocol, if a group of users want to securely talk with each other using IPSec, they must establish several IPSec point-to-point security association (SA). This approach is neither efficient nor scalable, and is not compatible with other multicast delivery services, such as IP multicast and Layer 2 switching (which relies heavily on multicasting and address filtering).

Virtual Bridged Local Area Networks (VLANs) [30] represent a standardized Layer 2 mechanism for partitioning a single physical LAN into multiple disjoint logical LANs, by assigning traffic to a particular VLAN based on the port number from which it arrived, or the value of the protocol/type field or an explicit tag carried within the packet. The IEEE 802.1Q standard [15] defines the packet format and required behavior for tagged-based VLANs. VLANs provide an efficient mechanism for improving performance by localizing traffic, i.e., reducing the span of a Layer 2 broadcast packet. Layer 2 switches are not supposed to relay traffic across VLAN boundaries; this requires a higher-layer policy decision by a router or other device. Nevertheless, tagged-VLANs by themselves cannot provide security because any single misconfigured (or compromised) switch could trivially compromise the separation between distinct VLANs [11], [27].

In addition to VLANs, several other Layer 2 approaches to security have been considered. Decades ago, the IEEE 802.10 standard [19] was created to define an interoperable security model among Layer 2 bridges. However, 802.10 is dead as it is too complex, and only Cisco still supports its format (but without encryption) for their FDDI switches (which are also dead). More recently, the IEEE 802 LAN/MAN Standards Committee has begun a new project on Link Layer security (LinkSec) [16]. However, LinkSec's charter is limited to providing one-hop link security, so their approach is not scalable: in order to communicate with a neighbor who is several hops away

we must trust every intermediate switch along the path. Finally, several vendors have developed proprietary security mechanisms. For example, Cisco has a mechanism in its switches to protect against certain types of Link Layer attacks, such as ARP spoofing [13].

B. Security Mechanisms with Host Participation

For security mechanisms needing end hosts' participation, we separate them to end-to-end and end-to-network, the latter is usually for network access. End-to-end security mechanisms pervade each network layer, such as S/MIME and PGP for email security over application layer, Transport Layer Security (TLS) and Secure Socket Layer on session layer (transport layer) and usually used for Web services. This kind of end-to-end application orient security mechanism is most secure according to end-to-end system design argument [29]. Much of the time, we also need a security mechanism to protect security-ignorant applications. IPSec transport mode is suitable to guard all specified application: it can combine with personal firewalls to provide a sound security service, and provides link-layer end-to-end security mechanisms as well [22].

IPSec end-to-network tunnel mode can be used to allow a single computer at some arbitrary remote location to establish a secure connection over the Internet to an enterprise gateway. More commonly, however, host-to-network security is more about blocking network access to non-authenticated devices, rather than protecting host traffic in transit across the insecure public Internet. In this case, network access is commonly handled by the IEEE 802.1X Port Based Network Access Control Standard [20]. For example, Cisco's Architecture for Voice, Video and Integrated Data (AVVID), integrates 802.1X with their proprietary Cisco Discovery Protocol to provide a mechanism to make the network more trustworthy: switches identify and authenticate their neighbors over the switch-to-switch links, while the edge switches authenticate the hosts.

802.1X defines a general framework by which a "client device", known as the *supplicant*, is authenticated by its "first point of attachment", i.e., the switch at the edge of the network known as the *authenticator*. 802.1X uses the Extensible Authentication Protocol (EAP) [6]. EAP can support multiple authentication methods and can work over Ethernet or wireless links.

Initially the supplicant's port is blocked except for relaying the EAP frames (i.e., authentication messages) between the supplicant and authentication server. Once authenticated, the controlled port is opened and all kind of frames are allowed.[1] This authentication process must be repeated each time the Ethernet physical layer transceiver reestablishes the link after a loss of carrier, even if the link is reserved for a single host (e.g., a staff person's desktop PC, or a networked printer). This re-authentication requirement is intended to prevent a network security breach if the host operating system is compromised, or if the network cable is moved from the usual host to an intruder's laptop computer.

PANS over CHOICE [4], NETBAR [8] and SPINACH [25] are all research projects related to wireless access, which is outside of scope of this paper. Note that the IEEE 802.1X standard has significant limitations in the wireless environment because the standard does not allow its authentication messages to be relayed over multiple links. Thus, 802.1X authentication cannot be used if multiple "dumb" access points are connected to a single intelligent switch to extend coverage. To solve this problem, a modification to 802.1X to support the relaying of authentication messages through a dumb wireless access point is being developed as security extension 802.11i [17] to the IEEE 802.11b Wireless LAN standard [18].

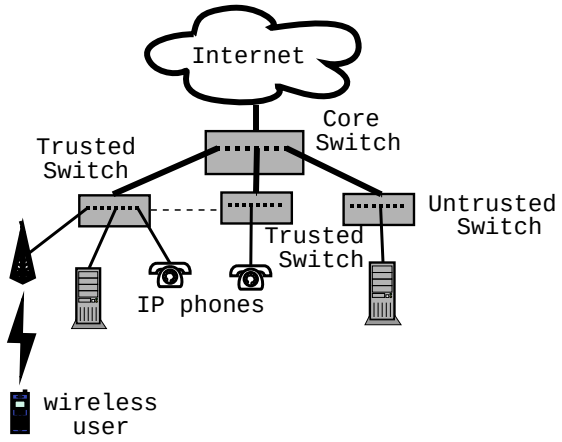

Fig. 1. S-VLAN architecture.

III. S-VLAN Architecture

In this section, we describe the main components of our application-based Secure VLAN (S-VLAN) architecture as shown in Figure 1, and their minimal requirements.

A. User Authentication at the Network Boundary

As we explained in Section I, we may not be able to trust the end hosts to properly authenticate users and to enforce access controls. An end host may be too "dumb"

[1] In some implementations, the client may be granted a user-specific set of network services, or perhaps assigned to a different VLAN.

to handle these tasks, because it is a simple embedded device (such as an IP phone) or its operating system does not support the concepts of resource protection and multiple users (such as a PC running Windows 9x, or a pre-OS X Macintosh). Furthermore, in visitor networks we must assume that users have complete priviledged access to all capabilities of their respective computers. Thus, our approach to user authentication should be host-independent and not depend on the integrity of the end hosts.

Clearly, the IEEE 802.1X Port Based Network Access Control Standard could be used for this purpose. More specifically, each network port remains blocked until the attached computer has been authenticated. In this case, however, we view port authentication as the equivalent to user login on a trustworthy computer. In other words, the switch knows that there is a process running on the attached computer, which has enough priviledges to generate network traffic, that has successfully authenticated itself as the given user id. We can now grant that host access to those remote resources which are available to that particular user — and no more, unless we trust the integrity of the end host operating system. Similarly, it would be unsafe to simultaneoulsy authenticate multiple users on the same host unless we trust its integrity.

In some cases, however, a trustworthy host may need to support multiple users at the same time, or to offer multiple services subject to different access control policies. In this case, we could apply VLAN tagging to the traffic on the link connecting such hosts to the trusted edge switch, in order to maintain the separation between users and/or services all the way to the host. For example, if multiple users share a trustworthy timesharing host, then the host would associate a distinct VLAN id to each user. Until the user authenticates itself to the edge switch, all traffic tagged with its VLAN id (and hence generated by the corresponding user) would be blocked at the switch.

Although the 802.1X standard has some known security flaws that could make it susceptible to a Man-in-the-Middle attack [2], we have developed a cross-layer position authentication mechanism in [28], which can be used to ensure that the integrity of the link is not compromised by the presence of intermediate devices between the supplicant and authenticator.

B. Trusted Edge Switches

A trusted edge switch is the most critical component in our architecture. As described in section III-A, the trusted edge switch authenticates each user before granting it access to the network and other remote resources. In addition, we assume that the switch is connected to the end host through a direct link that is capable of detecting loss of connectivity at the physical layer. This assumption holds for both the 100BASE-T Fast Ethernet and 1000BASE-T Gigabit Ethernet standards for transmission over balanced twisted-pair copper cabling, but is not valid for wireless networks. This is because the physical layer transceivers for high speed transmission over copper send a continous bi-directional stream of symbols over the link, whether or not their is any data being sent. Thus, if an authenticated host is unplugged from the link after gaining network access, and replaced by another host, the switch needs to know about the change and insist that the new host undergo a new authentication.

The trusted edge switch must also capable of applying the necessary resource allocation and access control policies to the network traffic associated with the services available to that host. Thus, we assume that the trusted switch can apply a multi-layer firewall access control rules to the traffic, which can be based not only on source or destination addresses (either MAC addresses or IP addresses or both), protocols, and even application-specific details such as port numbers. Some applications may be defined to use a fixed port number; for example, SSH uses port number 22. However other applications, including the VoIP H.323 standard, use dynamic port numbers [24]. In the latter case, the edge switch may need to support dynamic session-dependent policies. This could be accomplished by creating an application-dependent parser, such as the IP telephony parser for boundary firewall [26]. Conversely, the rules could be dynamically updated by some external agent, such as the VoIP Call Manager.

The edge switch enforces a set of rules that define a set of protected applications that are not openly available to all users. Other unprotected application traffic, such as Internet access, will go through regular default channel. By this method, the trusted edge switch can offer differentiated security services to specific applications.

C. Core Switches and Other Edge Switches

The only requirement for core switches is they need to be VLAN-awared and follow 802.1Q tagged VLAN standard. Such a feature is common for a middle-size switch acting as a core switch. So this requirement is reasonable.

Other edge switches, if they do not like to be one part of communication of the secure application, they do not need to change any functionality and they are not required to ensure any level of security. This is a much more relief compared with Cisco's AVVID. AVVID needs to upgrade every edge switch to be 802.1X enabled, and if one edge switch does not obey 802.1X standard, the security they claimed may not be achieved.

Weak security requirements for core switches and other edge switches can bring great benefits. If these switches are configured unproperly (this was claimed to be the most possible security problem) and have security holes, secured application still will not be hurted. A common scenario is at a large university, core switches are controlled by network administrators of the university, and edge switches are controlled by network administrators of different departments. Different department will have different security requirements. For example, computer science department wants to secure VoIP application, while library does not even want to use VoIP. In this case, admins from CS department can establish an application based S-VLAN overlay among their switches and do not need to worry about other edge switches not participating this application and also core switches. Thus, security problems such as VLAN hopping [31] will not happen in our S-VLAN.

IV. Secure VLAN Tunnels

If all users who are permitted to access a specific protected application were directly connected to a single trusted edge switch, then these dynamic multi-layer policy rules would be sufficient. However, this approach does not scale well if the users are distributed across multiple switches. For example, which switch should make the policy decision: the one connected to the client or the one connected to the server? Do all trusted switches need to know information about the policies applied to every user on every host?

To simplify the task of managing these policies, we assume that applications that include hosts connected to multiple trusted edge switches are linked together through encrypted VLAN tunnels between the trusted switches. In this case, if a host connected to trusted switch "A" wants to invoke a protected service to access a host on trusted switch "B", then we assume that switch "A" will first validate the local request and then transfer the traffic to the VLAN tunnel. Thus, when the traffic reaches switch "B" through the VLAN tunnel, it can be delivered to the destination without further validation because only trusted switches are able to use the VLAN tunnel. The design principles for these secure VLAN tunnels are a key part of our general approach and will be discussed in the following subsections.

All traffic on the VLAN tunnels carries a 802.1Q VLAN tag. The VLAN Number (VID) will be decided before hand, and each protected application will be assigned a different VID. That is why we call our approach an "application based" VLAN. Because of the VLAN tag, the packet can be switched transparently across the enterprise core network to the other trusted edge switches, assuming only that those core switches are VLAN-aware. Thus, our approach makes only minimal assumptions about the network core.

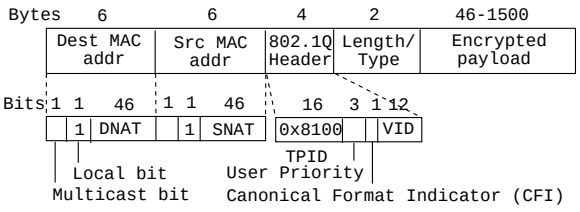

Fig. 2. S-VLAN packet format

The format of packets over S-VLAN tunnel is shown as in Figure2.

A. Encrypted Layer-2 Group Communication among Trusted Switches

In our approach, the entire Layer 2 payload (including IP Layer header) is encrypted throughout the entire path between the source and destination trusted edge switches. Thus, unlike IPSec (which only works for the IPv4 and IPv6 protocols), our security scheme is independent of the Layer 3 protocols: it works equally well for both standardized protocols (such as IP, and to a lesser extent IPX) as well as proprietary protocols for "dumb" embedded controllers that do not support IP, vendor-specific plug-and-play device autodiscovery and configuration protocols, network attached disks, etc.

In addition, since the standard Layer-2 transparent bridging and packet filtering algorithms do not care about payload encryption, our approach trivially generalizes to allow secure multipoint connections among multiple the trusted switches connected to the same VLAN. On the other hand, IPSec is inherently a point-to-point service, so group communications among N endpoints would need $O(N)$ distinct IPSec tunnels if multi-hop store-and-forward paths are allowed, or $O(N^2)$ distinct IPSec tunnels if we want a direct connection between each pair of switches. Furthermore, the switches would need to maintain routing tables that include how to reach each host, or at least which switch each host is connected to.

Since the Layer-2 payload is encrypted, all traffic on the secure VLAN tunnels is protected from disclosure to intruders, even while it is traveling through the (less secure) enterprise core switches. However, a malicious outsider could still launch a Denial-of-Service attack by inserting bogus traffic carrying the same VLAN tag through one of the core switches, or to attempt to disrupt some protected communications by a replay attack. These problems can be solved by augmenting the payload with a sequence

number field and integrity check before encryption, similar to IPSec. In the case where a service is defined by a particular port number, the sequence number and integrity check fields can be carried within the port number field in transit. Alternately, we could use the first 20 bits of the VLAN tag field for this purpose. In both cases, the length of the packet does not need to increase.

B. Layer-2 Network Address Translation

Network Address Translation (NAT) is normally associated with the IP layer. Layer-3 NAT is normally carried out at the boundary between some inner private network and the public Internet, either to allow multiple computers connected to the inner network to share a single IP address on the outside network, or simply to hide the addresses of computers on the inner network so they cannot accessed by outsiders. Layer-3 NAT can also be used to preserve the IPv4 address space, since the private addresses can occupy different address assignment ranges in IPv4 addresses family, or be reused between different private networks [12]).

Our purpose for introducing Layer-2 NAT is quite different. Since the Layer-2 address space is completely flat, and lacks any concept of locality and hierarchical structure similar to IP addresses, it is much harder to plug all the holes by which unauthorized traffic could reach a particular target. For example, access control lists (ACLs) are treated as a Layer-3 function in multi-layer switch routers, so these policy rules do not apply to traffic between two ports assigned to the same VLAN. Similarly, we have found some examples of VLAN-capable Layer-2 switches that share a single MAC address table across multiple VLANs, causing them to misdirect packets across VLAN boundaries or to allow a malicious user to disrupt the network connection for hosts connected to a different VLAN.

Thus, we use Layer-2 NAT as a means to create a single "choke point" where we can execute our policy rules, thereby preventing all contacts between the host (or protected service) and everything else running on the same enterprise LAN environment, unless it is compatible with our access control policies. Security breaches and bugs in one application will not affect the execution of other applications and the real machine. In other words, nobody should be able to reach target host (or protected service) without passing through our policy rules – no matter how close they are to the target host, or even if they know its correct MAC address (via "ARP", say). This approach is analogous to the virtual machine concept in operating systems, which is used to isolate the execution of one program from everything else, including the raw hardware.

In contrast to Layer-3 NAT, where the NAT gateway uses address multiplexing to share a single outside address among many internal hosts, in Layer-2 NAT we translate each host's MAC address to a distinct NAT-ed address, using a shared reversible function $fvid()$, which could be derived from the encryption key for the associated S-VLAN. Thus NAT-ed Layer-2 addresses are distinct from each other and from the actual MAC addresses for all hosts. Notice that, as part of our "virtual machine" style isolation philosophy, we translate *both* the source and destination addresses for *every* packet traveling within the S-VLAN. Furthermore, we use a shared reversible function to do the address translation, rather than some arbitrary mapping. In this way, the trusted edge switches determine the proper destination address for a packet without having to maintain a complete address mapping table for the entire S-VLAN.

Although IPSec tunnel mode could be used to provide equivalent functionality, it would require us to *encapsulate* all packets traveling through the tunnel in a new header, which would expand the size of each packet (i.e., a new MAC header adds at least 14 bytes), and restrict the tunnel's connectivity Conversely, the standard Layer-2 transparent bridging and packet filtering algorithms continue to function correctly for S-VLAN traffic, even when it is traveling through "other" switches that know nothing about the S-VLAN concept except the establishment of the appropriate forwarding rules for the associated VLAN id. In other words, if host A wants to send a packet to host B, the edge switch adjacent to A will encrypt the packet and translate the source and destination addresses to $fvid(A)$ and $fvid(B)$, respectively. If the intermediate switches don't recognize these translated addresses, they will broadcast the packet and remember through which port they can reach $fvid(A)$. Otherwise, the intermediate switches will simply forward the encrypted packet across the S-VLAN to the edge switch adjacent to B. Similarly, when such an encrypted packet reaches a trusted edge switch, it filters, forwards or decrypts and delivers the packet to a local host as necessary.

The Layer-2 address translation function takes advantage of a special bit called "local bit", which is included in the IEEE MAC address format [14]. Setting the local bit to 1 means we have created a privately administered MAC address. Thus, our Layer-2 NAT function $fvid()$ must leave multicast-bit unchanged, and force the local bit to be 1. Furthermore the NAT-ed address must be unique in this VLAN and, to avoid confusing the core switches, it should be unique in the whole LAN. This goal can be achieved by letting the $fvid()$ depend on original MAC address and carefully chosen cryptographic functions. And how to de-

sign a good NAT function family still remains as open issue. One possible effort will be setting a master function F, then using F to derive a family of slave functions so that each function corresponds to aVLAN, thus the finally NAT-ed address will be a function of (F, VLAN number, original address) and be unique in the whole LAN.

The L2-NAT acts as one more layer of access control security protection and is not redundant. Without layer 2 NAT, an attacker can add bogus VLAN header to its packet and send bogus packets to a trusted edge switches. The trusted edge switch will finally discard the packet once it figures out the payload is not legitimate after decrypting L2 payload and do integrity check and anti-replay check, but these operations will waste a lot of its capabilities. If L2-NAT is not used, an attacker can easily exploit this and launch Denial of Service attacks against trusted edge switches. However, if we use L2-NAT, once the trusted switch find a packet with the S-VLAN number but wrong MAC address, it will discard the packet and save its effort. This means that L2-NAT makes our system more robust to DoS attacks.

C. Bootstrap, Key Agreement and Management

Key agreement and management can use out-of-band control channel rather than data channel among switches, similar to bridge management data unit BPDU [21]. Bootstrap can be done manually or automatically before using the secure application. For example, if CS department wants to use VoIP over S-VLAN, the administrators first ask for a 802.1Q VID from the university administrators. The VID is known to core switches in the control region of the university admins. Then, edge switches participating secured VoIP (and allowed as entry switches for VoIP) negotiate key related dtails, such as key exchange algorithms, master keys, session keys (may derived from master keys). Usually master keys are seldom change, while session keys can be dynamically change. The frequency of re-key depends on time, packets sent out, etc, and is still an open problem to us.

V. S-VLAN Implementation

We implemented a proof-of-concept testbed for our application based Secure VLAN architecture design. We used Linux computers to represent the end hosts, although any network devices could have been used.

We used some desktop PCs running Linux Mandrake 9.1, kernel 2.6.0 with bridging enabled and built-in layer 2 firewall (ebtables) support, to represent the trusted edge switches. We extended ebtables as a muli-layer firewall and revised one of ebtables targets to do S-VLAN processing as described above. We emulate a manually bootstrap process. The encryption algorithm we chose is AES [9], as it is suitable for hardware implementation later on real switches.

Of course, for real implementation, it would be better to use a commercial switch with reprogramming capability, such as the Nortel Passport 4000, in combination with hardware support for encryption/decryption at the ports. We hope to study this approach in our future work.

We used an Extreme Networks Summit 48i to represent the core network. This is a high performance commercial switch with 48 ports and support for 802.1Q VLAN-tagging.

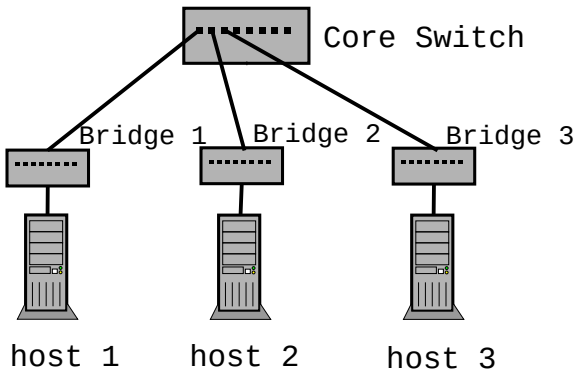

Fig. 3. Test Bed.

VI. Experiments and Results

A. Conformance Tests

The testbed is shown as Figure3. We changed ebtables rules and functions, so after ebtables filtered which packet can go to S-VLAN, the packet is NAT-ed, inserted VLAN header, and encrypted, and then sent through the core switch to the other edge trusted switch. Once it reached there, it is DNATed, stripped VLAN header, and decrypted, and then sent to the correct end host.

We tested both applications such as ssh and ping which use static port numbers and application such as VoIP (we use ohphone under linux) which use dynamic port number for data transport. For VoIP packets, we used a range of port numbers. More sophisticated alternatives will be using VoIP parser as described in [26].

For these applications and during the test procedure, we used ethereal [10] to monitor traffic over the links of the tunnel and could only see the changed packets. Ethereal could only tell such packets were 802.1Q VLAN packets. The source MAC addresses, destination MAC addresses and the content of L2 payload were all kept as secrets.

To test whether our mechanism works under attacks, we connected a third end host to an untrusted edge switch then to a port of the core switch. This "attacker" tried to use the secured applications, say ssh. The effort failed. We did a portmap to the second host on the third host, and could see "filtered" for ssh while open for other unprotected applications. As a comparasion, if the first host portmapped the second host, it would see "open" for ssh. This means the access control scheme works well. The user cannot use a secured application through connecting to an untrusted edge switch (switch that does not participate to the S-VLAN).

We also configured the untrusted switch to be on the S-VLAN (having the same VID) and got the same results as above. This means misconfiguration on core switches is tolerable.

B. Performance Tests

We used ohpone under Linux on two end hosts to gather statistics of VoIP calls. We compared the round trip time, number of packets dropped, number of packets out of order and number of packets arrived late and did not see much difference.

VII. Conclusions and Future Work

We designed an application-based secure VLAN architecture and implemented a prototype system. Our application based S-VLAN architecture is suitable for hot spots, university and enterprises LANs and can efficiently prevent inside security problems from interrupting secured applications. Our Application based Secure VLAN (AS-VLAN) is simpler and more efficient than IPSec in LAN environment as it conforms to Layer 2 sematics, and it is cost efficient to real-time applications such as VoIP.

Our future work includes implement the real AS-VLAN architecture in commercial programmable switches and move crypto algorithms to be implemented by hardware to achieve wire-speed performance, and to integrate VoIP parser to the multilayer firewall.

Acknowledgement We gratefully ackowledge TCS funding support for this research. We also thank Professor Chinya Ravishankar for his helpful comments.

References

[1] Alcatel. Authenticated VLANs. White paper, Alcatel, November 2002.

[2] W. Arbaugh and A. Mishra. An initial security analysis of the IEEE 802.1X standard. http://www.cs.umd.edu/ waa/1x.pdf, June 2002.

[3] R. Atkinson. Security architecture for the Internet Protocol. IETF RFC 2401, November 1998.

[4] P. Bahl, A. Balachandran, and S. Venkatachary. Secure wireless internet access in public places. In *Proc. ICC 2001*, Finland, June 2001. IEEE.

[5] S. Bellovin. Distributed fi rewalls. *USENIX ;login*, pages 37–39, November 1999.

[6] L. Blunk and J. Vollbrecht. PPP Extensible Authentication Protocol (EAP).

[7] CAIDA. Packet length distributions, 2000.

[8] CMU. NetBar - Carnegie Mellon's solution to authenticated access for mobile machines.

[9] J. Daemen and V. Rijmen. AES proposal: Rijndael. http://citeseer.ist.psu.edu/daemen98aes.html, 1998.

[10] *Ethereal: A Network Protocol Analyzer*. http://www.ethereal.com.

[11] R. Farrow. VLAN insecurity. http://www.spirit.com/Network/net0103.html.

[12] A. A. for Private Internets. Address allocation for private internets, Mar. 1994. IEFT RFC 1597.

[13] C. Howard. Layer 2 – the weakest link. http://www.cisco.com.

[14] IANA. Ethernet vendor address components or organizationally unique identifi ers. http://www.iana.org/assignments/ethernet-numbers.

[15] IEEE Computer Society. IEEE 802.1: 802.1Q - Virtual LANs. http://www.ieee802.org/1/pages/802.1Q.html.

[16] IEEE Computer Society. IEEE 802.1 link security task group. http://www.ieee802.org/1/pages/linksec/.

[17] IEEE Computer Society. IEEE 802.11, Task Group I. http://grouper.ieee.org/groups/802/11/Reports/tgi_update.htm.

[18] IEEE Computer Society. IEEE 802.11, The Working Group for Wireless LANs. http://grouper.ieee.org/groups/802/11/.

[19] IEEE Computer Society. IEEE Std 802.10, IEEE Standards for Local and Metropolitan Area Networks, 1998.

[20] IEEE Computer Society. IEEE std 802.1x-2001, Port Based Network Access Control, 2001.

[21] IEEE Computer Society. IEEE 802.1: 802.1D - MAC Bridges, 2003. http://www.ieee802.org/1/pages/802.1D-2003.html.

[22] R. Khoussainov and A. Patel. LAN security: Problems and solutions for ethernet networks. *Computer Standards and Interfaces*, 22:191–202, 2000.

[23] D. Leifer. Visitor networks. *the Internet Protocol Journal*, 5(3):2–16, September 2002.

[24] McClellan Consulting. Layer 4 through Layer 7 switching. Technology white paper, McClellan Consulting. http://www.mcclellanconsulting.com/whitepapers/Layer4through7.pdf.

[25] E. Poger and M. G. Baker. Secure Public Internet Access Handler (SPINACH). In *Proc. USENIX Symposium on Internet Technologies and Systems*, December 1997.

[26] B. Reynolds. Enabling secure ip telephony in enterprise networks. Master's thesis, UC Davis, Dec. 2002.

[27] S. Rouiller. Virtual LAN security: Weaknesses and countermeasures. Technical report, SANS Institue, 2003.

[28] A. Saha and M. Molle. Thinking outside the box: Extending 802.1X authentication to remote "splitter" ports by combining physical and data link layer techniques. In *Proceedings of 28th Annual IEEE International Conference on Local Computer Networks*, Germany, October 2003.

[29] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-End arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, Nov. 1984.

[30] R. Seifert. *The Switch Book: The Complete Guide to LAN Switching Technology*. John Wiley & Sons, June 2000.

[31] @stake. Secure use of VLANs: An @stake security assessment. Research report, @stake, August 2002.