SANE: A Protection Architecture For Enterprise Networks

Obi Akonjang
(oa2k@hotmail.com)

Seminar "Internet Sicherheit" ,
Technische Universität Berlin

WS 2006/2007 (Version of January 20, 2007)

Abstract

In a relatively short period, enterprise networks have evolved from small-sized LANs with simple architectures, to present day large networks with very complex architectures. Their topologies now include combinations of Local Area Networks (LANs), Wireless access networks, Metropolitan Area Networks (MANs), Wide Area Networks (WANs) and Virtual Private Networks (VPNs) that often span across multiple continents. Sustaining these highly interconnected, but also more dynamic and complex architectures currently occurs through the implementation of complex routing and switching protocols. Nevertheless, the increased network connectivity and higher availability have still not been sufficiently balanced by improved security. Threats from Viruses, worms, trojan horses and DoS attacks are still persistent, with rising tendencies in their sophistication and ability to spread. Mechanisms such as ACL, packet filters, firewalls, IDS and IPS, etc, put in place to curb these increased levels of threats and attacks have also caused the network to become inflexible, fragile and difficult to manage.

This paper addresses issues such as trust, access control, complex routing and switching, and other forms of attacks that affect present day enterprise networks. It evaluates and analyzes current methods used to resolve these issues, points out their limitations and then proposes a new approach in dealing with the fundamental problem. It presents a newly designed protection architecture (SANE) for the enterprise network. This architecture is based on a single, logically centralized protection layer that is used to setup, secure and control all connectivities within the network.

1 Introduction

When the Internet was originally designed, viruses and other forms of malicious attacks were quite uncommon. As a result, little or no efforts were made to include security in the original design. Much importance was placed on factors such as universal connectivity, decentralized control and openness. These factors contribute to the rapid growth and huge successes of the Internet, and have also enormously contributed to the widespread adoption and use of TCP/IP, the Internet-based protocol stack. The is also the *de facto* protocol stack used in enterprise networks.

Nevertheless, the openness, rapid growth and ever-increasing popularity of the Internet also make it a preferred medium for the easy launch and spread of viruses, worms, malwares, and other forms of attacks. These attacks do not only stay within the bounds of the Internet, but also extend into the enterprise network, often causing increased loss and damage to enterprise assets and leading to business interruptions, regulatory exposures, brand damages, lost productivity and lower profitability [5]. The importance of security in an enterprise network can thus no longer be neglected. Mechanisms such as router Access Control List (ACL), firewalls, Network Address Translations (NATs) , Virtual Local Area Networks (VLANs) and many other middleboxes have thus been added to the enterprise network architecture to address these issues.

The implementation of these mechanisms has in effect increased the complexity of the enterprise network and reduced its openness. Trading openness for security has still not done enough to solve the problems [4][7]. These mechanisms still require huge amounts of configuration and oversight, are often limited in the range of policies they can enforce, produce complex and fragile networks and have still not significantly improved the security within the enterprise. Worms are still causing losses in productivity and potential for data loss. Attacks resulting in the theft of intellectual property and sensitive information are still very common.

The lack of an appropriate solution and the need to correct a fundamental problem, motivated the authors of this paper to start all over from scratch and redesign a new network architecture, with security being a fundamental goal [1]. The result of their efforts is the *Secure Architecture for the Networked Enterprise (SANE)*. Its main design goals are:

- establish an architecture that supports simple but still very powerful natural policies that are independent of both topology and equipment used,
- implement security at the link layer, thereby preventing more permissive lower layers (usually the link layer) from undermining controls implemented at upper layers (usually from the network layer upwards),
- hides all topology and services information from unauthorized parties and
- have only one trusted component within the network, instead of relying on distributed trust amongst multiple components, which usually leads to increased risk potentials.

The rest of this paper is structured as follows; Section 2 examines current enterprise security mechanisms and points out their limitations. Section 3 describes the SANE architecture in more detail. Section 4 shows how this new approach successfully addresses the limitations mentioned in Section 2. A prototype implementation and evaluation of a SANE network is presented in Section 5. Conclusions and my personal opinion are presented in the Section 6.

2 Current Solutions and their Limitations

Many of the approaches used so far to tackle the issues within the enterprise network have mostly concentrated on secluded aspects, such as simplifying complex routing designs [8], enhancing routing protocols and router securities [13] and using hybrid trust models to increase flexibility [2] and ease administration [10]. Despite their successful implementations, these methods have still not been able to provide the needed solution because they have mostly addressed only smaller fractions of the broader problem, leaving other aspects unresolved.

Another approach involves the creation of distributed security policies using several mechanisms such as VLANs, ACLs, firewalls, NAT, IDS, IPS, etc that all run on separate systems. Configurations become quite complex using this method and the security also becomes fragile. A work-around had been to put all of these into a single box at a choke point in the network and create a single policy out of them. But then, an attacker that successfully breaks in has free access to rest of the internal network.

In [11], it is argued that current models only temporarily solve immediate security problems. Underlying weaknesses still persist, leaving the network open to attacks. They further state that a new model has to be developed, or significant changes incorporated into current architecture if the security gab is to be closed. Such was the effort made in developing and implementing the IPSec standard, which nevertheless, still lacks in features and has compatibility issues [12]. Thus, the approach of adjusting original design deficits by building and improving on previous works [7], does not always lead to a satisfactory resolution of the fundamental problem. Starting all over from scratch with a clean slate [1] is yet another approach, but rarely put to use. The SANE network is designed using this approach.

3 SANE Architecture

SANE uses two approaches to ensure that security policies are enforced at the link layer during all end host communications. The first approach (clean-slate approach) modifies all Network components to support SANE and the second enables inter-operation between unmodified end hosts running the standard IP stack and SANE-enabled end hosts or the DC.

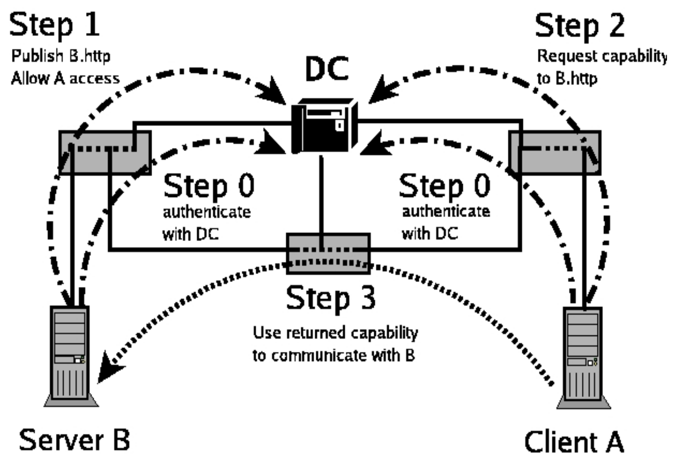

Figure 1: *The SANE Service Model showing the steps host A must take to establish connectivity with Server B; Step 0 - Authentication phase. All principals authenticate to DC and establish secure communication links; Step 1 - Server publishes its hosted service using a unique name.; Step 2 - Client A obtains capability from DC to access service hosted by Server B; Step 3 - Client A contacts Server using acquired capability.*

3.1 The Domain Controller (DC)

The Domain Controller is the logical core of a SANE network. Its main functions include

- authentication of users and end hosts,
- advertisement and control of connections to available services and
- issuance of capabilities (encrypted source routes) to hosts. Capabilities control how hosts communicate within the SANE network environment.

The DC uses four separate service modules to implement these functions. These include the authentication service module, the network service directory module, the topology service module and the capability construction service module. The topology and capability service modules are both parts of the Protection Layer Controller, examined in a separate section below.

3.1.1 Authentication Service Module

All entities within a SANE network must be authenticated before they are allowed to take part in any form of communication. This service module is used to authenticate switches and principals (users and hosts) within a SANE network. The Authentication Service module establishes and maintains a symmetric key with each entity that it uses to securely communicate with each of them.

3.1.2 The Network Service Directory (NSD) Module

The NSD is responsible for maintaining a hierarchy of directories and published services within the SANE network. It publishes these services using unique names, such as *S.ftp*, which indicates a server S that offers ftp service. The NSD stores separate access control lists (ACL) of each service and directory that states which users or groups have rights to view, access, modify or publish a particular service or directory. It also does names resolution, just like the Domain Name Service (DNS) in a normal IP network. If a principal needs to access a particular service, it first checks with the NSD. The NSD then goes through the particular ACL for that service to see if the principal is allowed to access it. If it is permitted to access the requested service, the NSD will resolve the name, get the DC to issue a capability and send back the issued capability to the requesting principal.

3.2 The Protection Layer

In the **clean-slate approach**, a new (SANE) header (see Figure 2 below) is created between the Ethernet header and the IP header, for all packets.

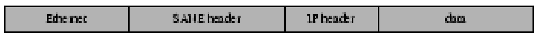

Figure 2: *The SANE header found in each packet in a SANE network. It is used to establish the paths that a packet is permitted to use.*

In SANE, all initial connectivity attempts are directed per default to the DC. A connection can then be established by building a minimum spanning tree (MST) between the host (switch) and the DC (root of the tree), an approach which is similar to the distance vector approach, used in ethernet switches.

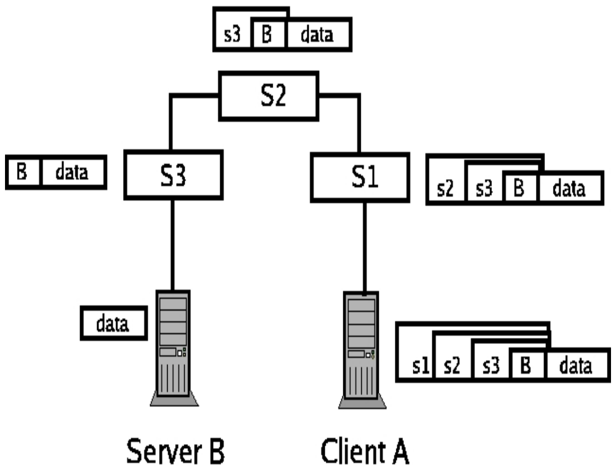

Figure 3: *Switching in a SANE network. Forwarding packets from client A across switches S1, S2, and S3 to Server B.*

The MST algorithm has a property that prevents switches from learning the network topology or reproducing it using packet traces. The only use of the spanning tree is to create default routes that will be used to forward packets to the DC.

When a switch sends a request to the DC to request a capability, the DC must be able to communicate back to the switch, in order to establish symmetric keys. These are needed for authentication and also for generating and decoding capabilities. Capabilities are normally encrypted in layers (see Figure 3 above) to hide the topology and also prove that they originated from the DC. In order to establish a full topology of the network, the DC initially starts by establishing shared keys with its immediate neighbors and then iteratively contacts switches that are further away, until a complete trust system is built. The established shared keys provide confidentiality, integrity and defence against replay, for all subsequent traffic between the DC and the switch by means of an *authenticator* header.

The Protection Layer Controller is responsible for keeping a complete topological view of the network and computes routes when necessary. It is also responsible for restoring the network to a functioning state if things do go wrong and can get rid of misbehaving switches by instructing its immediate neighbors to drop all traffic from such switches.

All capability requests and link state updates are sent via the MST to the DC. As packets are sent along the MST, switches construct their *request capability* by generating encrypted onion at each hop containing the previous and next hop. The Dc uses these request capabilities to communicate back to each sender and also to determine the location of all senders, since paths are also encoded in the capabilities.

Hosts communicate by means of *FORWARD* packets that carry the capability provided by the DC. Once this packet arrives at a switch, the switch first checks if the capability is still valid (i.e. if it has not expired or been revoked) before letting it through.

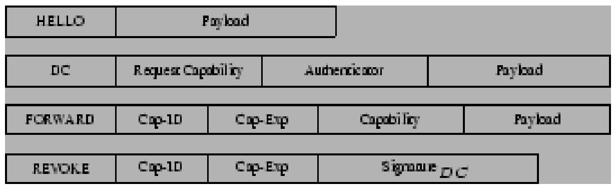

Figure 4: *Types of packet in a SANE network.*

If a misbehaving sender tries to use a capability on a victim, the victim can send a revocation request to the DC, consisting of the final layer of the offending capability. The DC checks if the requester is on the capability's path and returns a signed packet of type *REVOKED*. The requester then forwards it upstream to the switch from which the misbehaving capability was forwarded, traveling hop-by-hop along the reverse path of the offending capability. The switches along this path then verify the DC's digital signature, if valid, they then add the revoked CAP-ID to their local revocation list and use it to compare against each incoming packet. The switch will drop any match and forward the revocation to the previous hop. Since a revocation only makes sense during the lifetime of its corresponding capability, both carry the same expiration time.

Interoperability with unmodified end hosts is the second approach used by SANE. In this case, *translation proxies* are used to map standard IP events to SANE events. *Gateways* are used to provide the same translation service for wide-area connectivity.

Translation proxies are the very first hops that packets from all unmodified end hosts get sent to. They translate between IP naming events and SANE events, mapping DNS queries to DC service lookups and the corresponding DC lookup replies back to DNS replies. They handle capabilities on behalf of the end host by caching those issued by the DC, attaching them to all outgoing packets from the host. The same method is used to translate other discovery protocols such as SLP, DNS SD, and uPNP that are currently being used in today's enterprise networks. For full compatibility, SANE translation proxies must be able to handle all service lookups and translate all queries and replies that are sent to and received from the DC respectively.

Gateways are placed on the perimeter of a SANE network and do function as perimeter NATs between the SANE network and the WAN. They cache the capabilities of outgoing packets and generate a map of capabilitiy to corresponding IP packet header. They use it to check against incoming packets, appending the capability in case of a match, before forwarding the packet.

Broadcast is used by some discovery protocols (e.g. uPNP) to perform service discovery on the LAN. To achieve this, they broadcast lookup requests to all hosts within the LAN, violating the least privilege principle implemented by SANE. To support broadcast, SANE forces all link layer broadcasts to be sent to the DC, from where they are safely and appropriately forwarded to all other hosts. All replies are sent back to the DC, which then returns them to the sender. The DC thus acts as the sole channel through which all broadcast requests and corresponding responses are sent, and can thereby verify that they strictly conform to established specifications. Being the sole channel also allows the DC to cache learned services and use them for subsequent requests, thereby reducing the total amount of broadcast traffic within the LAN.

Service Publication within SANE can be done in a number of ways. Services are published through the DC, either by translating existing service publication events, using a command line, via a web interface or through *binding* on the local host (sockets) in case of IP.

3.3 Fault Tolerance

Although the DC is logically centralized, it could be physically replicated to increase scalability and provide fault tolerance. Switches can then separately authenticate and connect to multiple DCs via multiple spanning trees, sending their neighbor lists to each of them. Each DC, being the root of their corresponding tree, can thus grant routes independently of each other without any need for a topology consistency between them. The selection of a DC to send requests to is done in a random manner by each host, enabling the distribution of traffic.

SANE leaves the responsibility of determining network failures to the end host, because a direct communication between a switch and the end host will be in violation of the least privilege principle and could create means for DoS attacks. By sending periodic probe or keep-alive messages, SANE aware end-hosts can detect failures in the network and request new capabilities. A link failure causes the DC to be flooded by requests from parties willing to re-join the network. Clients will be able to adapt quicker if the DC is able to issue multiple edge-disjoint capabilities. This enables clients to quickly use another capability to rejoin the network.

3.4 Additional Features

Middleboxes and Proxies. In a standard IP network, proxies can only be placed at particular points, but with a SANE network, it could be placed anywhere determined by the DC. Capabilities could be used to enforce user- or service-specific policies.

Client Mobility within the LAN is transparent to servers because the underlying topology is transparent to services.

Anti-mobility. With SANE, it is possible to prevent hosts and switches from moving by blocking access to them if they do. This could be used to deny access to rogue PCs, but it could also be a hindrance in a highly mobile environment.

Centralized Logging. The DC, being the pivot of all communications could be used as a centralized logging system and also for forensics.

4 Preventing and Resisting Attacks

The design of SANE to effectively centralize control on the DC, conceal networking topology and strictly implement the policy of least privilege helps exclude many of the vulnerabilities that are present in today's networks.
SANE resist common attacks in the following ways:

- The NSD uses ACLs to control access to services and directories, thereby preventing unauthorized discovery and listing of services in the system.
- SANE uses only encrypted source routes and encrypted link-state updates to communicate. This prevents spoofing and any sort of man-in-the-middle attack.
- Only authenticated network components are allowed to communicate in a SANE network. The authentication mechanism prevents unauthenticated switches from joining the network. Spanning-tree and routing attacks are impossible because of the DC's central role.

Resisting more sophisticated attacks demands more from SANE. It turns to degrade when faced with such attacks. The major classes of these attacks are further examined in the subsequent sub-sections.

4.1 Resource Exhaustion

Flooding attacks: The DC uses rate-limits and revocation to handle flooding attacks. Any end device (switch or host) that violates this limit gets disconnected from the network.

Revocation state exhaustion: Switches in SANE networks keep a list of revoked capabilities. If an attacker stores up many capabilities and then cause all of them to be revoked simultaneously, the list could then easily fill up if stored in e.g. a small Content Addressable Memory (CAM). To prevent such attacks, the switch generates a new key when it notices that its revocation list is full and invalidates all existing capabilities passing through it. It then clears the filled list before sending the new key to the DC.

On the other hand, the DC also keeps track of all revocations per sender. If a sender exceeds a predefine threshold for a particular service, the sender is removed from that service's ACL.

If a misbehaving switch uses a sender's capability to flood a receiver, the sender can cut off the switch by simply using a different capability and then bypass the misbehaving switch. Well-behaved switches would then need to use or request an alternative capability and also bypass the misbehaving switch, which could cause temporary performance degradation. With this approach, SANE networks can quickly converge to a state, in which attackers hold no valid capabilities and are also blocked from obtaining new ones.

4.2 Malicious Switches

Although switches in a SANE network have only minimal functionality by design and require authenticated public keys to join the network, malicious switches can still use other methods, such as hardware tampering or supply chain attacks to gain access to the network and disrupt normal network operations.

Disrupting MST Discovery: A malicious switch can cause additional DC traffic to pass through it by falsely advertising a better distance to the DC during MST build-up. This can cause traffic inefficiency or worse even, degrade throughput if the switch instead attracts and drops packets. A more refined attack, where the malicious switch selectively drops packets from a particular neighbor is illustrated in the Figure 5.

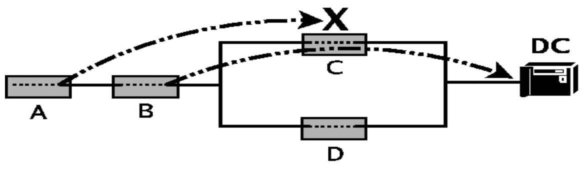

Figure 5: *Malicious switch C selectively drops packets from A while letting those from B through.*

Since only packets from Node A get dropped and not those from Node B, Node B maintains its path via the malicious switch C, and forwards all packets it receives via C, including those from A, thereby preventing A from communicating with the DC, although an alternative path via D still exists that could be used.

Such attacks can be prevented by concealing the sender's identity from intermediate switches. Other methods include:

- hiding easily recognizable sender-IDs from packet headers,
- hiding the path length of all response capabilities by padding them to the same length,
- hiding a node's scheduled timings by randomizing its periodic messages to the DC.

With these measures in place, any (malicious) switch that drops most of the packets through it will force its immediate neighbors to reconstruct the MST excluding it. In case of occasional drops, the MST discovery could be temporarily degraded, switches down the line will, nevertheless, still be able to register with the DC.

Bad Link-State Advertisement: If a malicious switch succeeds to disrupt or divert normal traffic flow by using falsified information in its link-state updates, it could temporarily cause a denial-of-service attack. This would only be temporary because SANE enables end hosts to request for fresh capabilities immediately they realize they are unable to use the false route.

4.3 Malicious Domain Controller

In a SANE network, the DC is the most important entity and is highly trusted. The compromise of this entity would render full control of the network to the attacker. To avoid it being a single point-of-failure and prevent an attacker from assuming control of it, multiple DCs could be used instead, such that trust is distributed amongst them (instead of lying on a single one) using e.g. threshold cryptography. With this method, the DC's secret key is stored in multiply locations, across multiple DCs (e.g. n $<$ 4), such that a minimum multiple (e.g n=2) of them are needed to generate a capability. Senders would then need to communicate with 2 out of 4 DCs to obtain a capability. The same mechanism can also be used to prevent a single malicious DC from revoking arbitrary capabilities.

5 SANE Implementation and Evaluation

A SANE prototype implementation was carried out in [1] using switches, IP proxies and a DC. The network environment also included Seven 100Mb Ethernet workstations with their maximum transmission unit (MTU) reduced to 1300 bytes to create room for the SANE header. The Virtual Network System (VNS) that has the ability to specify and test complex topologies was also used.

5.1 SANE Switches and IP Proxies Implementations

Unmodified end hosts are supported by using proxy elements (developed for SANE) that are placed between the hosts and their first hop switches. These proxies use ARP cache poisoning to redirect all traffic from the end hosts and to cache their capabilities. These capabilities are inserted into in-coming packets from the end hosts before being propagated into the SANE network and removed from out-going packets before being sent to the unmodified end hosts.

The switches use automatic neighbor discovery to discover their neighbors and exchanged HELLO messages every 15s seconds with them. They also support MST construction, link-state updates and packet forwarding. The switches are also able to reconfigure the MST and update the DC whenever they encounter a link failure. Each switch keeps a table of individual capabilities revocations that contains *Cap-IDs* and their corresponding expiration times. This is the only dynamic state maintained in the switches. The Capabilities were created using OCB-AES and decrypted with 128-bit keys. Using OCB both confidentiality and data integrity are achieved with just a single pass over the data, generating a ciphertext that is exactly 8 bytes longer than the input plaintext.

5.2 Domain Controller Implementation

The public keys of all the switches are pre-configured on the DC and used for authentication. In constructing capabilities, end-to-end path calculations are made using searches in both directions, from the source and to the destination and vice versa. All computed routes are also cached at the DC and used to accelerate subsequent capability requests (but only after it had been checked against the current topology to ensure its correctness).

The DC hosts a web-server that provides an HTTP interface to the service directory. All DNS queries from unauthenticated users resolve to the DC's IP address. With a browser and the right permissions, users are able to log on to the web-server, browse the service directory and carry out other activities, such as adding a new service. The directory service also has an interface for user and group management.

Users access services by browsing through the directory tree and selecting the desired service link. Their permissions are then checked and if successful, the DC generates a capability and sends it to the client. The web-server then returns an HTTP redirect with the service's right protocol and IP address, e.g. *ssh://192.168.120.5:22/.*

5.3 Evaluation

By first studying the performance of the software implementation of the switches and the DC (see sections 5.1 and 5.2) and comparing these with those obtained from packet traces in a medium-sized enterprise network, the authors in [1] were able to analyze the practical implications of running SANE on a real networking environment and further address scalability and DC replication concerns.

	5 hops	10 hops	15 hops
DC	100,000 cap/s	40,000 cap/s	20,000 cap/s
Switch	762 Mb/s	480 Mb/s	250 Mb/s

Table 1: *DC and Switch performance*

Testing with a 2.3GHz PC, the performance of the DC (in capabilities per second) and switches (in Mb/s) for different capability sizes were measured as shown in **Table 1** above.

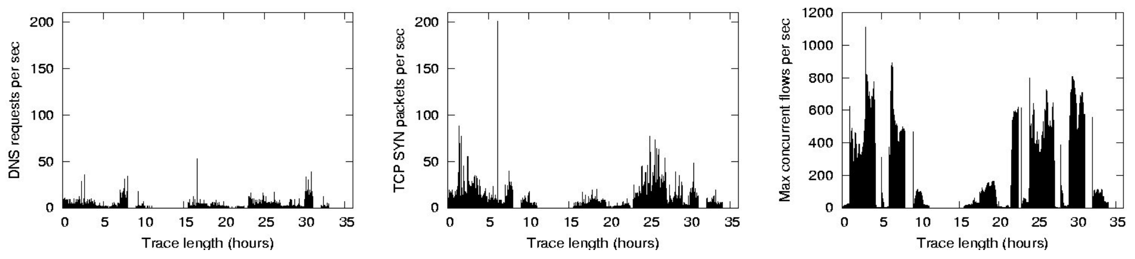

Figure 6: *Measured DNS requests, TCP connection establishment requests and maximum concurrent TCP connection per second respectively.*

To analyze the issue of scalability and DC replication, traffic traces from a medium-sized enterprise, involving about 8000 internal addresses, 47 million packets (including 20,849 DNS and 145,577 TCP connections) were collected over a period of 34 hours. The traces apply anonymization techniques to ensure consistencies in the mappings between hosts' real IP addresses and the published anonymized ones and also to ensure the preservation of real port numbers, which are used to identify the application-level protocols of the packets.

The DNS request rate, TCP connection establishment rate and the maximum number of concurrent TCP connections per second, are respectively shown in Figure 6. The first two rates give an estimate of what is to be expected in a SANE network. As can be seen, the upper bound of the TCP connection request is approximately 200 requests per second, i.e. about 200 times lower than that used in the DC implementation (see Table 1).

A link failure might cause all hosts using the link to simultaneously requests new capabilities from the DC. In the case of the traces, 1,111 is the observed maximum number of concurrent connections that would be affected. Assuming this number represents a worst case scenario, it was used to estimate the bandwidth consumption of control traffic in a SANE network, result from e.g. a total link failure. Using header sizes of the prototype and assuming a maximum path of 10 hops, the packets carrying forward and return capabilities would be at most 0.4Kb in size and thus produce only a maximum of 0.646Mb/s of control traffic.

This demonstrates that a few DCs are capable of handling large numbers of requests from tens of thousands of end hosts simultaneously. On the other hand, DC replications are actually relevant, if service interruptions are to be avoided when an active DC fails.

6 Conclusion

This paper addressed some of the common issues affecting trust, access control, connectivity and security in today's enterprise networks. It analyzed some of the methods being used to deal with them and pointed out their limitations and drawbacks. Using a clean-slate approach, it showed how a new centralized architecture (SANE) could be used to avoid, prevent and even resolve most of the identified issues.

The simplicity of the SANE approach is a big plus in easing the administration and control of large enterprise networks.

Despite greatly simplifying the network architecture on the one hand, the adopted centralized approach also pulls away valuable processing resources, on the other hand, from otherwise robust switches and router to a rather busier DC, which could at any time, be a potential bottle-neck or a single source of failure, i.e. even if only one of its offered services fails or malfunctions, while the rest run correctly. This is also possible in the case of replicated DCs.

With all routing within a SANE network being determined by the centralized DC alone, it seems, SANE can only be suitable for networks with flat architectures. On the contrary, most large enterprise networks have hierarchical architectures and use hierarchical routing algorithms to mimic these structures. Using SANE on such networks might instead raise more issues than it might help resolve them.

Testing SANE on a software switch in a test environment revealed that a high percentage of the CPU (99%) was spent decrypting. This was however attributed to an unoptimized encryption, which nevertheless led to a poor throughput. This nevertheless still poses a potential weak point when faced with of a DoS attack.

Avoiding or disconnecting malicious switches and reconstructing new capabilities could be good on the one hand when resisting attacks, but would disrupt or negatively affect the quality of time-sensitive services such as Voice over IP (VoIP) communications on the other hand, if capabilities have to be newly requested.

Although the restrictive nature of a SANE network helps enhance its security, it also takes away the flexibility and openness that are common in an enterprise network.

SANE does not address (and potentially can't support in its present form) some other key component of present day enterprise networks, such as the implementation of Virtual Local Area Networks (VLANs). It still needs to prove its effectiveness and robustness under harsher conditions, such as when faced with Distributed Denial of Service (DDOS) attacks, affecting multiple switches and multiple end hosts simultaneously.

References

[1] M. Casado, T. Garfinkel, A. Akella, M.J. Freedman, D. Boneh, N. McKeown and S. Shanker. SANE: A Protection Architecture for Enterprise Networks, In 15th USENIX Security Symposium (Security '06).

[2] Barbara L. Fox and Brian A. LaMacchia, Cooperative Security: A Model for the New Enterprise, Proceedings of the Seventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '98), Stanford, CA, June 1998, 314-319.

[3] Jovanovic, N.; Sorgic, D.; Tianying Ji; Shaowen Song; An overview of metropolitan and enterprise networks - current and future, Canadian Conference on Electrical and Computer Engineering 2005. 1-4 May 2005 Page(s):160 - 163.

[4] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet Denial-of-Service with Capabilities. SIGCOMM Comput. Commun. Rev., 34(1):39-44, 2004.

[5] The Open Network Problem http://www.esecurelive.com/

[6] Vandenwauver, M.; Claessens, J.; Moreau, W.; Vaduva, C.; Maier, R.; Why enterprises need more than firewalls and intrusion detection systems: Infrastructure for Collaborative Enterprises, 1999. (WET ICE '99) Proceedings. IEEE 8th International Workshops on Enabling Technologies 16-18 June 1999 Page(s):152 - 157

[7] Haji, F.; Lindsay, L.; Shaowen Song; Practical security strategy for SCADA automation systems and networks; Canadian Conference on Electrical and Computer Engineering, 2005. 1-4 May 2005 Page(s):172 - 178

[8] G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, and G. Hjalmtysson: Routing design in operational networks: A look from the inside. In Proc. ACM SIGCOMM '04, pages 27-40, New York, NY, USA, 2004. ACM Press.

[9] Kiely, L.; Benzel, T.V.; Systemic Security Management; Security and Privacy Magazine, IEEE Volume 4, Issue 6, Nov.-Dec. 2006 Page(s):74 - 77

[10] Maley, J.G.; Enterprise security infrastructure: Infrastructure for Collaborative Enterprises, 1996. Proceedings of the 5th Workshop on Enabling Technologies 19-21 June 1996 Page(s):92 - 99

[11] Kuper, P.; The state of security; Security and Privacy Magazine, IEEE Volume 3, Issue 5, Sept.-Oct. 2005 Page(s):51 - 53

[12] Berger, T.; Analysis of current VPN technologies; The First International Conference on Availability, Reliability and Security, ARES 2006. 20-22 April 2006 Page(s):8 pp.

[13] Dijiang Huang, Qing Cao, Amit Sinha, Marc J. Schniederjans, Cory Beard, Lein Harn, Deep Medhi; New architecture for intra-domain network security issues; Communications of the ACM, November 2006, Volume 49 Issue 11