L3A: A Protocol for Layer Three Accounting*

Alwyn Goodloe, Matthew Jacobs and Gaurav Shah
University of Pennsylvania

Carl A. Gunter
University of Illinois Urbana-Champaign

November 2005

Abstract

Accounting protocols are used to quantify traffic to support billing, QoS assurances, and other objectives. Current protocols do not provide complete security for this purpose because of the threat of 'cramming' attacks in which unauthenticated parties can introduce traffic that the accounting system attributes incorrectly. In this paper we explain this vulnerability and introduce a protocol, Layer Three Accounting (L3A), that addresses it through the coordinated establishment of a family of IPsec tunnels. Our goal is to give a practical specification and implementation of the protocol and show its efficiency. We demonstrate that the latency for setting up and tearing down L3A connections is about one third slower than one gets for end-to-end connections alone, but the bulk rate of transfer is improved by 100% over the typical alternative configuration for accounting.

1. Introduction

Accounting protocols are used to quantify traffic to support billing, QoS assurances, and other objectives. A common way to do this, as seen in cellular data systems, is to associate traffic with specific clients on an access network and use a Network Access Server (NAS) with an associated accounting system such as a RADIUS server to maintain accounting records such as the number of bytes or packets the client exchanges with a server on the Internet. This is done by authenticating the client to the NAS by the use of a cryptographic tunnel authenticated with a credential from the client, or by cruder means, such as associating the client to a specific MAC or IP address. This approach provides some level of secure attribution of the traffic that passes from the client across the accounting node into the Internet. Response traffic that is directed to the client is also attributed as part of the accounting system. However, this response traffic is less securely identified than the client's traffic to the NAS since it is not explicitly authenticated as valid response traffic at the accounting node. This raises a threat that an adversary in the Internet will 'cram' traffic into the authenticated channel between the NAS and the client and such traffic will be attributed to the user as valid response traffic. If the user is properly authenticating his connection to the server in the Internet, then such traffic will be discarded, but not before it has been attributed to his account by the accounting node, which did not perform a similar authentication.

The aim of this paper is to explore an approach to preventing such attacks by authenticating traffic across the accounting node in both directions, including traffic sent *to* the client as well as traffic sent *from* the client. Specifically, we consider how to do this with a protocol that coordinates a collection of network layer tunnels. It is common to provide accounting through the use of link layer mechanisms, especially for wireless links, but a network layer solution offers advantages for portability. The primary contributions are a protocol, L3A, for layer three accounting that addresses cramming attacks and a demonstration that the protocol can be efficiently implemented using a family of IPsec tunnels. Excellent progress has been made in recent years on the Internet Key Exchange (IKE) protocol for dynamic tunnel establishment between pairs of nodes. The next step to leverage this progress is to show how IKE can be used as building block in more complex multi-node protocols to achieve high-level security objectives such as robust accounting. The design and analysis of such protocols can be subtle. In this paper we focus on practical aspects such as the specification of a protocol that is simple to implement with satisfactory efficiency using existing standards and software. In particular, we have specified L3A and implemented it on FreeBSD based on our own implementation of (a fragment) of IKEv2. Our experiments show that L3A accounting costs about 160ms for both set up and tear down. This is about 2.4 times more than the same operations for an IPsec tunnel alone. However, tunnel reuse in L3A reduces this to a factor of 1.5 in the common case where the client-to-NAS tunnel already exists. On the other hand, L3A improves bulk traffic performance by 100% over a naive (but typical) approach to accounting where accounting uses an encrypted tunnel to the NAS.

In the second section we provide background on cramming attacks, our network layer approach, and related work. The third section describes the L3A protocols. The fourth section describes our implementation and experiments. The fifth section concludes.

2. Background

Some background is required to understand cramming attacks and our approach to solving them. We begin with a de-

*In: Workshop on Secure Network Protocols (NPSec), November 2005, Boston, Massachusetts.

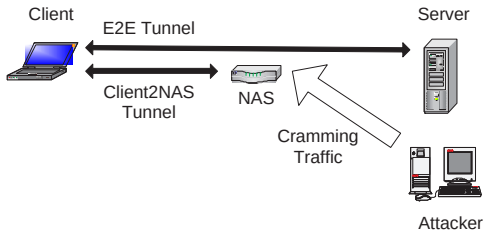

Figure 1: Cramming Attack

scription of the problem, then we sketch our network layer approach to solving it, and then survey related work.

2.1. Cramming Attacks

Figure 1 illustrates a typical example of a packet-based communication link with an accounting element. The NAS is placed at a network bottleneck, such as a wireless access point or router, where it monitors traffic to and from the clients who will be charged for network access. The NAS is typically supported by an authentication and accounting system such as a RADIUS server and collects information about such parameters as throughput of the client, the number of sessions it runs, the duration of its access, or anything else it is able to record. To ensure proper attribution, a tunnel can be placed between the client and the NAS so that each packet from the client is authenticated. Such tunnels are often placed at link layer, but could be placed at virtually any network layer. The client uses its connection through the NAS to visit various sites in the Internet where it finds servers. Clients often secure the link to the server with an additional tunnel, which stretches end-to-end between the client and the server through the NAS. This tunnel provides privacy from, among other things, the NAS itself. As the client makes requests to the server and the server sends its responses through the NAS, the NAS does its accounting.

The architecture of Figure 1 suffers from a gap in its protection of the NAS accounting system. The NAS is able to authenticate all traffic coming from the client and will (typically) drop traffic it receives from any other source on the client-side interface that purports to come from the client but is not authenticated. By contrast, response traffic from the server is unauthenticated by the accounting system. This raises a threat that a node on the Internet could direct false response traffic into the NAS. Since the NAS does not authenticate traffic on the server-side interface, it will typically dispatch this traffic on to the client. The client will probably discard the traffic since it will not match its tunnel to the server, but by the time this traffic reaches the client, it has been attributed to it by the NAS thus compromising the integrity of the accounting database. We refer to this as a *cramming attack*.

The actual details of the cramming attack depend on the network architecture and details of the accounting mechanism. The seriousness of the threat depends on how response traffic is forwarded to the client by the NAS. For networks that use globally routable network IP addresses and allow arbitrary services to be run on the clients (*i.e.* outside hosts can initiate connections to these services), the cramming attack is easy to perform in the absence of additional firewalling mechanism at the NAS. Firewalling on the client side may still be in use, but as the NAS will not be aware of it, it will forward any packet (and account for it) onto the client host.

If Network Address Translation (NAT) is being used, a cramming attack is more complicated. For purposes of this discussion we only consider the TCP protocol: details for UDP are similar. As NAT is used to share one globally routable address with hosts having private addresses, an incoming packet will only be forwarded if the NAT router determines it to be a part of an existing connection initiated earlier by a client. Connections are are identified by a 4-tuple which response packets must use. As the destination IP in the response packets will be that of the NAS address, three remaining values need to be determined by the attacker: the IP address of the server, the server port number, and the client port number. Guessing these values for a particular client's connection is challenging for an off-path attacker (that is, an attacker that is not on the routing path between the NAS and the server).

However, some tricks can be used to make the network vulnerable to cramming attacks against random clients. For networks that support a large number of clients, many users are likely to be connected to relatively popular services on the web such as popular search engines and portals (*e.g.* `google.com`), instant messaging services (*e.g.* AIM), IMAP and POP mail access (*e.g.* `gmail.com`, `yahoo.com`), and so on. Thus, the attacker has a large number of fixed server IP/port pairs to choose from as possible endpoints for different connections a NAT NAS might be tracking. Only the client port information needs to be guessed. Client ports are often chosen from a fixed set of ranges (ephemeral port ranges) whose exact values are dependent on the particular OS and configuration. By using different client port ranges and sending out packets with different client port values picked from probable ranges, it is possible to get response packets past the NAS and hence successfully perform the cramming attack. There are some NAT implementations that make this a very effective approach. For instance, if port numbers are allocated sequentially and there is an insider behind the NAS, then active port number can guessed easily. Even if there is no such edge, a brute force attack can achieve some success. On a Pentium 4 running Linux 2.6.10 at 2.4 Ghz with 1GB of RAM we were able to send packets with a 1.4Kb payload and varying port values at a rate of around 10,000 packets/second with code that had no optimizations and no changes to the drivers or the kernel. Thus, a brute force attack on the client port numbers can be performed in a small amount of time. Although we do not know a specific way to do it, the existence of any technique for telling if a cramming packet 'hit' a client would make such an attack quite effective.

The time window in which the attack is successful depends on the length of the time period for which the NAT router maintains state information for each connection. As this state information is the one that is used to ascertain whether to forward a certain response packet to a client, it is maintained for at least the period of the connection. For connections that only last momentarily (*e.g.* HTTP), it is important that the attack take place in the period when the NAT router still has this state in-

formation stored. RFC 2663[18] recommends that the NAT router maintain state for at least another 4 minutes (2 * Maximum Segment Lifetime) after it thinks that the connection has terminated. As a NAT router can never be sure whether the connection tear-down packets it saw on the wire actually reached the destination host, it continues to forward packets for that connection for a little while after the observed teardown (to enable retransmissions). Our experiments with a Linksys wireless router doing NAT showed that connection state was maintained for 7 more minutes after the actual connection was terminated. Thus it was possible to cram packets into this connection after its actual termination for another 7 minutes. These factors contribute to increasing the length of the vulnerability time window and hence give plenty of time to the attacker for a brute force attack.

2.2. Layer Three Accounting

These points above show that the most assured approach to preventing a cramming attack is to establish some kind of tunnel between the NAS and the server with parameters that are not vulnerable to guessing, even by an on-path attacker. A simple approach would be for the NAS to set up the tunnel when it sees a communication request from one of its clients. This can be done at a variety of network layers, possibly using higher-layer protocols like SSL or SSH.

Our approach with L3A is to coordinate the establishment of all three of the tunnels involved in this protection system and base them on IPsec. IPsec [12] is a suite of network-layer security protocols that has been standardized in the IETF. Cryptographic tunnels called *security associations* (SA) define the transformations that get applied to each packet traveling in the association. Although unidirectional, they are usually created in pairs with one association flowing in each direction. Each node maintains a security association database (SADB) containing information on the associations active at that node. *Security policies* determine which packets travel in which security association. Each node maintains a security policy database (SPDB) that contains the policies that apply to incoming and outgoing traffic. Though they use classical cryptographic protocols as building blocks, L3A and its sibling tear-down protocol can be viewed as signaling protocols. That is, they install and manipulate state in the network comprising the association and policy databases. To avoid manual key configuration, the symmetric keys used by IPsec security associations are usually established by invoking a key-exchange protocol. The IETF designed the Internet Key Exchange (IKE) [11] protocol for just this purpose. IKE not only establishes the shared key, it sets up a pair of associations between the two parties.

Figure 2 illustrates the basic configuration, where each of the black double-headed arrows indicates an IPsec connection esbtablished using IKE and with corresponding entries in the SADB and SPDB at each end. Having a unified network-layer multi-tunnel protocol has at least three advantages. First, the tunnels can be coordinated for efficiency. The two NAS tunnels can do only authentication since they were intended only for accounting while the end-to-end tunnel provides confidentiality. Second, it is possible to develop a suitable credential mechanism for the overall protocol rather than just for each individual

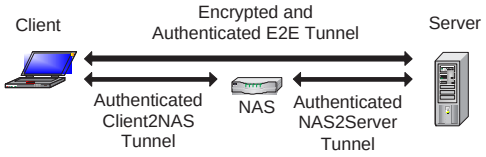

Figure 2: Layer Three Accounting Tunnels

tunnel. For instance, the NAS is able to use an L3A credential from the client to establish its connection to the server. Third, an implementation at network layer can exploit the elegance and advances of IPsec. For instance, IPsec provides robust DoS protection, has hardware support, and is portable across many link layers.

2.3. Related Work

There are two main areas of related work for L3A: protocols for accounting and protocols for dynamic establishment of IPsec tunnels.

There are many proposed schemes proposed for accounting for wireless services [4, 9, 13]. In order to create reliable bills, the accounting system must be secure and reliable. There are several accounting protocols in use today. Among them are Simple Network Management Protocol (SNMP) [1] and the RADIUS accounting protocol [16]. Although they may require a client to authenticate themselves, they do not protect against the cramming attacks described above. We are not the first to identify the billing and accounting systems as a ripe target for attacks. Since a company may lose customers or face costly financial credits due to over-billing, there are obvious incentives to pay for protecting accounting services and this fact has not been lost on the vendors of security devices. Network security products are now being advertised as providing protection against such attacks. For example, Juniper Networks claims that their security gateway for GPRS provides protection against over-billing attacks [10]. The security devices being aimed at this market are generally sophisticated stateful firewalls and we have seen that cramming attacks can evade such measures.

The tools available today for the management of IPsec tunnels remain relatively limited. Even when using IKE to dynamically establish tunnels, there must be some prior configuration of each node. Network managers usually do this from the router's command line interface. Centralized management tools such as Solsoft's Policy server [17] provide a network manager the capability of configuring IPsec tunnels from a central location. Fu and Wu have proposed a centralized management system that generates the detailed IPsec policy and association entries from high-level specifications [6, 7]. Centralized configuration is impractical in our scenario given that the NAS would have to be preconfigured to connect to all servers that a client can access. Others have developed protocols to set up IPsec connections. Cisco's Dynamic Multipoint VPN (DM VPN) [2] feature is a protocol for establishing tunnels between spoke nodes of a hub and spoke configuration. Each spoke node must maintain a permanent tunnel a to a Next Hop Resolution

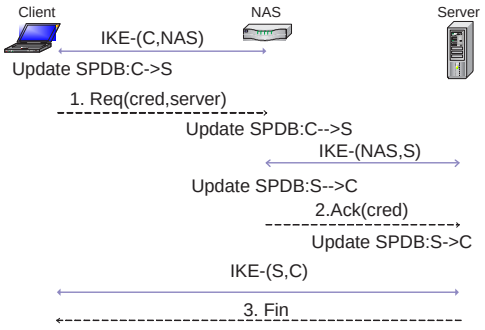

Figure 3: L3A Set Up

Protocol (NHRP) [14] server acting as a hub. When a spoke wishes to communicate directly with another spoke, it queries the hub for the information it needs to set up a tunnel. Cisco's Tunnel Endpoint Discovery (TED) protocol [5, 3] is protocol for discovering the endpoint of a tunnel and setting up the association.

3. Protocols

In this section we present the two protocols for layer three accounting. The L3A set up protocol was described formally in [8] using term rewriting. This section provides a specification aimed at implementers and also provides an L3A tear down protocol.

3.1. L3A Set Up

The L3A set-up protocol creates the configuration of tunnels given in Figure 2. The client will initiate the protocol. Although the client has a relationship with the NAS as well as with the server, the NAS is not assumed to have any relationship with the server. So the client must pass a credential to the NAS that it presents to the server on behalf of the client in order to establish the tunnel between the NAS and the server. IKE will be used to as a building block to set up the three pairs of tunnels. Additional messages are required to pass credentials, to tell the NAS which server it should set up a tunnel with, and to tell the client that the protocol has completed. The protocol is illustrated in Figure 3 and the details are as follows.

Client initiates protocol The client C identifies the server S it desires to contact and the NAS that will provide access to the Internet.

Establish Client-NAS SA If there does not already exist a pair of associations between C and NAS, then the client invokes IKE to establish a pair of associations between the client and the NAS.

The client updates its SPDB with a policy saying that all traffic from the client to the server should flow through the $\mathsf{C} \rightarrow \mathsf{NAS}$ association. The client then forms a message containing the name of the server with which the client wishes to communicate and a credential that the NAS presents to the server on behalf of the client.

Msg1 $\mathsf{C} \rightarrow \mathsf{NAS} : \mathsf{Req}(\mathrm{cred}, \mathsf{S})$

NAS Receives Req If the NAS receives notification that a key exchange has completed with the client, it waits to receive a Req. On the other hand, a Req message may be received in a preexisting association. When the Req(Cred,S) message arrives, the NAS updates its SPDB with a policy saying that all traffic from the client to the server should flow through the $\mathsf{C} \rightarrow \mathsf{NAS}$ association. The NAS extracts the server address and credential from the Req message.

Establish NAS-Server SA If there does not already exist a pair of associations between NAS and S, then the NAS invokes IKE to establish an SA between the NAS and the server.

The NAS updates its policy database to reflect the fact that all traffic flowing from the server to the client should travel in the $\mathsf{S} \rightarrow \mathsf{NAS}$ association. The NAS then sends the message

Msg2 $\mathsf{NAS} \rightarrow \mathsf{S} : \mathsf{ACK}(\mathrm{cred})$

Server Receives Ack Upon notification that a key exchange with the NAS has occurred, the server waits for an Ack message to arrive in the newly created tunnel. On the other hand, the Ack may arrive in a preexisting tunnel. Upon receiving the Ack message, the server extracts the credential that had been passed by the client. If the credential is valid, the server updates its SPDB with a policy saying that all traffic flowing from the server to the client should travel in the $\mathsf{S} \rightarrow \mathsf{NAS}$ association.

Establish Client-Server SA The server invokes IKE to set up a pair of end-to-end associations between the S and C. When the key exchange is complete the server notifies the client that the protocol has completed by sending a message.

Msg3 $\mathsf{S} \rightarrow \mathsf{C} : \mathsf{FIN}$

3.2. L3A Tear Down

We now give a brief description of the protocol that tears down the tunnels established by L3A. The steps are illustrated in Figure 4. We label the associations a-f as in the figure. In practice these identifiers would be the SPIs for each association. The protocol works as follows.

Client initiates protocol The client initiates the protocol by sending to the server

Msg1 $\mathsf{C} \rightarrow \mathsf{S} : \mathsf{delete}(e)$ Upon receiving a message of this form, the server deletes the association and policy for e. and sends the message

Msg2 $\mathsf{S} \rightarrow PC : \mathsf{delete}(e, f)$ and removes the association and policy for f. Upon receiving a message of this form, the client deletes the e and f associations as well as their policies. The client then sends the message

Msg3 $\mathsf{C} \rightarrow \mathsf{NAS} : \mathrm{TDReq}(\mathsf{C}, n, s)$ Upon receiving a message of this form, the NAS sends a message

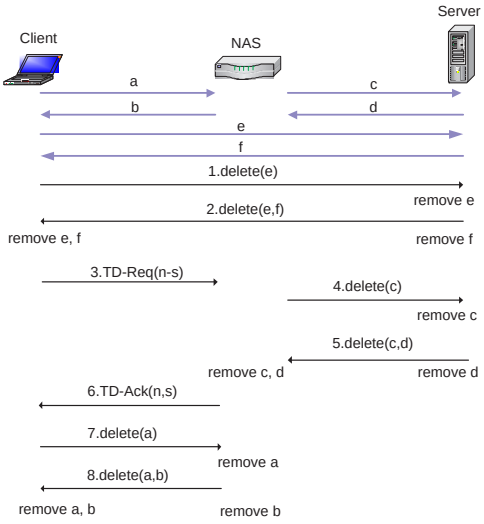

Figure 4: L3A Tear Down

Msg4 NAS $\rightarrow$ S : delete(C, c) Upon receiving a message of this form, the server removes the policy for all traffic flowing from S $\rightarrow$ C. If there are no polices for other clients, the association is deleted. The server sends the message

Msg5 S $\rightarrow$ NAS : delete(C, S, c, d) If the c association was removed, the d association is now removed. Upon receiving this message the NAS removes the policy for S $\rightarrow$ C and removes c and d if the tunnel is not in use by another client. The NAS then forms the following message

Msg6 NAS $\rightarrow$ C : TDAck(n, s) Upon receiving a message of this form, the client checks if there are sessions with other servers and if not, then the client forms

Msg7 C $\rightarrow$ NAS : delete(a) Upon receiving a message of this form, the NAS deletes the association and policy for e and sends the message

Msg8 NAS $\rightarrow$ C : delete(a, b) and removes the association and policy for b. Upon receiving a message of this form, the client removes the a and b associations as well as their policies.

4. Implementation

In order to demonstrate the practicality of the L3A protocol, an actual system running L3A is needed. We implemented the three principals of L3A (client, NAS, server) on three different machines, each running FreeBSD 4.8 and connected with a megabit/second network link. In our testbed, the client and server both are Micron 600MHz Pentium IIIs with 128MB of memory, and the NAS is a Dell 1.3 GHz Pentium IV with 256MB of memory.

The cryptographic operations of L3A are performed using the standard OpenSSL library. Updates to the kernel's SADB made by L3A are communicated through the PF_KEY Key Management API, as described in RFC 2367[15]. Interestingly,

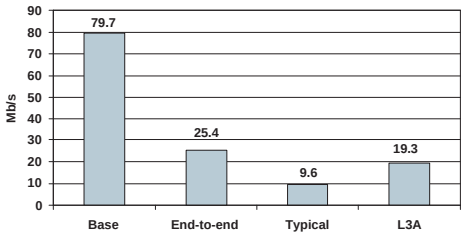

Figure 5: Throughputs

this RFC does not completely describe the manipulation of the SPD, so a trial and error approach was required in order to interface with L3A correctly.

Our biggest challenge in the implementation was getting a suitable implementation of IKE. We wanted to use IKEv2 but with support for nested tunnels on our FreeBSD platform. Unfortunately we did not find an off-the-shelf system satisfying these requirements so we set out to implement enough of it ourselves to do our experiments. The IKE protocol is rather complex in that there are many terms of negotiation to support the most general use. Such complexity is not needed for our purposes. Instead, we employ a somewhat stripped down version of IKE that we call IKE-. The simplifications are achieved by assuming that encryption is done with triple DES and that authentication is done using HMAC rather than negotiating which cryptographic mechanisms to use. A special flag is included in our protocol to indicate if the resulting tunnel should perform encryption and authentication or only authenticate traffic. Another simplification comes from eliminating IKE's two phase structure, where child associations are created in a second phase based on the shared key established in the first. IKE- has a single phase that terminates with the establishment of associations flowing in both direction. An appendix gives details of the IKE- protocol.

The experiments run on our testbed were designed to give performance results of the L3A protocol that can be compared to the performance of other solutions to the accounting problem. The first set of tests measures the raw throughput of client-server communication in 4 cases. The results appear in Figure 4. The first case, Base, is a baseline, with no IPsec at all, in order to quantify the maximum possible throughput of the connection. The second, End-to-end, utilizes IPsec end-to-end encryption and authentication, without accounting guarantees. Encryption has a significant impact on throughput, reducing it to barely a third of the unencrypted rate. This is particularly pronounced in the third case, Typical, which is the configuration of tunnels seen in most current accounting systems. The client maintains an encrypted tunnel with both the NAS and the server. This double encryption degrades performance to about a third of the End-to-end case because of the double encryption burden it places on the client. The final case, L3A, uses the tunnels set up by L3A. This entails three tunnels rather than the two used in the Typical case, but encryption is performed only end-to-end and the cost of the authenticated tunnels is much less that that of those that apply encryption. Consequently, the throughput performance of L3A is 101% better than that of the

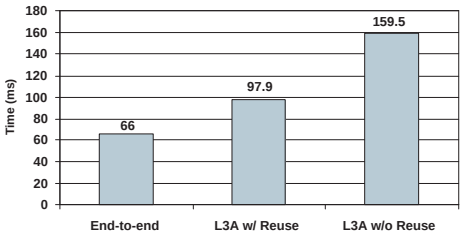

Figure 6: Latencies

Typical configuration and only 32% lower than the case with no accounting. We did not measure the case where there are large numbers of clients, but in our tests with one client, the NAS was only lightly loaded.

The second set of tests measures the latency of the tunnel setup times. The results appear in Figure 4. For each scenario, the data reflects the time taken to both set up and tear down all appropriate tunnels. In the simplest case, End-to-end, just the client-server tunnel is set up. The next case to consider, L3A w/ Reuse, occurs in the common case, where a client-NAS tunnel already exists, and the remaining NAS-server and client-server tunnels are created and torn down by L3A, leaving the client-NAS tunnel for another L3A session. The final case, L3A w/o Reuse, describes the latency of L3A when the client-NAS tunnel is not reused, and all three tunnels are created and torn down by L3A. Thus the latency cost of establishing tunnels for accounting is 142% greater than that of end-to-end protection alone, but in the most common case, when there is already a tunnel between the client and NAS, it will be only 48% longer.

5. Conclusion

We have described cramming attacks and shown they pose practical threats to existing accounting protocols. We have described a way to address these threats by the coordinated establishment of a collection if IPsec tunnels using a protocol for layer three accounting called L3A. We implemented the L3A protocol on FreeBSD machines using our own implementation of a fragment of IKE and given evidence that the performance of L3A provides gains in bulk efficiency over more naive approaches while costing relatively little in increased latency. Perhaps the main contribution of the work is the demonstration of an elegant and efficient protocol for the dynamic establishment of a coordinated family of IPsec tunnels to achieve a multi-domain security objective (*viz.* secure accounting).

Acknoledgements. We would like to thank Tom Hiller and Pete McCann who first suggested we investigate the problem of securing the wireless accounting infrastructure. We would also like to thank Steve Bellovin for useful comments during several discussions on this work. This research was supported by the Office of Naval Research under MURI Research Contract N00014-02-1-0715.

References

[1] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple Network Management Protocol. RFC 1157, IETF, 1990.

[2] Dynamic Multipoint VPN (DM VPN). Cisco White Paper. `http://www.cisco.com/`.

[3] Tunnel Endpoint Discovery Enhamcement. Cisco White Paper. `http://www.cisco.com/`.

[4] J. Cushnie and D. Hutchison. Charging and Billing Challenges for Wireless Internet Networks. Technical report, Lancaster University.

[5] S. Fluhrer. Tunnel Endpoint Discovery. Technical report, IPSP, 2001. `http://quimby.gnus.org/internet-drafts/draft-fluhrer-ted-00.txt`.

[6] Z. Fu, S. Wu, W.H. Haung, K.Loh, F. Gong, and C. Xu. IPsec/VPN Security Policy: Correctness and Conflict Resoloution. In M. Sloman and J. Lobo and E. Lupu, editor, *Policies for Distributed Systems and Networks*, pages 39–56, 2001.

[7] Z. Fu and S.F. Wu. Automatic Generation of IPSec/VPN Security Policies in and Intra-Domain Environment. In *International Workshop on Distributed Systems: Operations and Management*, Lecture Notes in Computer Science 1995, pages 279–290, October 2001.

[8] Alwyn Goodloe, Mark-Oliver Stehr, and Carl A. Gunter. Formal simulation of layer 3 accounting. In *Workshop on Issues in the Theory of Security (WITS '05)*, Long Beach, CA, January 2005. IFIP.

[9] Evolution of Charging and Billing Models for GSM and Future Mobile Interner Services, 2000.

[10] Infrastructure Security Gateway for GPRS Networks. Juniper Networks White Paper. `http://www.juniper.net`.

[11] C. Kaufman. Internet Key Exchange(IKE V2) Protocol. RFC 2407, IETF, 2004.

[12] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, IETF, 1998.

[13] M. Koutsopoulou, A. Kaloxylos, A. Alonistioti, L. Merakos, and K. Kawamura. Charging, Accounting, and Billing Management Schemes in Mobile Telecommuncations Networks and the Internet. *IEEE Communications Surveys*, 6(1), 2004.

[14] J. Luciani, D. Katz, D. Pisciello, B. Cole, and N. Doraswamy. Nbma next hop resolution protocol (nhrp). RFC 2332, IETF, 1998.

[15] D. McDonald, C. Metz, and B. Phan. PF_KEY Key Management API, Version 2. RFC 2367, IETF, 1998.

[16] C. Rigney. RADIUS Accounting. RFC 2866, IETF, 2000.

[17] Solsoft Policy Server: Better Management for Network Security. Solsoft White Paper, 2003. `http://www.solsoft.com`.

[18] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, IETF, 1999.

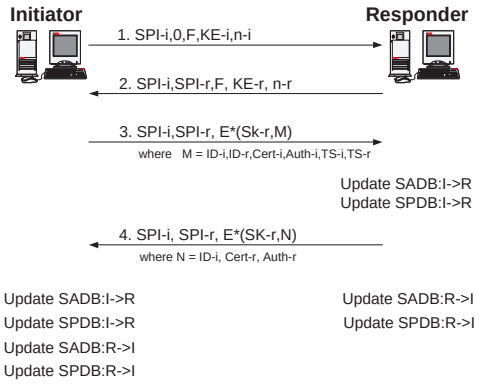

Figure 7: IKE-

A IKE-

The basic idea behind the IKE protocol is that an initiator and a responder perform a Diffie-Hellman key exchange to establish a shared key. This key will be used to generate a pair of symmetric keys for authentication and encryption. These keys are used to establish a pair of unidirectional associations between the two nodes. Upon termination, the association and policy databases have been updated accordingly. IKE- is an implementation of a stripped-down version of IKEv2 that supports nested tunnels.

Let K be a symmetric key. We write $S(K, M)$ for a signature function (such as HMAC) and $E(K, M)$ for an encryption function (such as triple DES). Assuming K_a and K_e are keys for authentication and encryption respectively, we abbreviate:

$$S^*(K_a, M) \stackrel{\text{def}}{=} M, S(K_a, M)$$
$$E^*((K_a, K_e), M) \stackrel{\text{def}}{=} S^*(K_a, E(K_e, M)).$$

The IKE$-$ protocol is illustrated in Figure 7. Details of the protocol description are as follows.

Initiation The protocol has two principals: an initiator I and a responder R. Principal I generates a nonce n_I, a SPI value SPI_I for the $\mathsf{R} \rightarrow \mathsf{I}$ association, a flag F indicating if the resulting tunnel is to perform encryption and authentication or just authenticate, and the Diffie-Hellman value KE_I. The initiator then sends the message

Msg 1 $\mathsf{I} \rightarrow \mathsf{R} : \text{SPI}_\mathsf{I}, 0, \text{F}, \text{KE}_\mathsf{I}, n_\mathsf{I}$

If R gets a message of this form, it generates a SPI value SPI_R for the $\mathsf{I} \rightarrow \mathsf{R}$ association, a nonce n_R, and a Diffie-Hellman value KE_R. The responder then sends the message

Msg 2 $\mathsf{R} \rightarrow \mathsf{I} : \text{SPI}_\mathsf{I}, \text{SPI}_\mathsf{R}, \text{F}, \text{KE}_\mathsf{R}, n_\mathsf{R}$

Generate Keys Both sides can now generate SKEYSEED from which all the cryptographic keys for the resulting SAs are derived. Separate keys for authentication and encryption are computed for both directions. These keys are known as SK_e^I, SK_a^I, SK_e^R, and SK_a^R for the encryption and authentication of the initiator and responder tunnels. When missing the subscript, SK^I and SK^R denote a pair of authentication and encryption keys.

The initiator now fetches its certificate Γ_I. It also generates

$$\text{Auth}_\mathsf{I} = S(SK_a^\mathsf{I}, (\text{Msg1}, n_\mathsf{R}, S(SK_a^\mathsf{I}, \text{ID}_\mathsf{I}))),$$

where ID_I is the identity of the initiator. This proves the initiator's knowledge of the secret corresponding to ID_I and integrity protects the contents of the first message. The policy selectors for the resulting SAs are also generated. The initiator then sends the message

Msg 3 $\mathsf{I} \rightarrow \mathsf{R} : \text{SPI}_\mathsf{I}, \text{SPI}_\mathsf{R}, E^*(SK^\mathsf{R}, M)$

where $M = (\text{ID}_\mathsf{I}, \text{ID}_\mathsf{R}, \Gamma_\mathsf{I}, \text{Auth}_\mathsf{I}, \text{TS}_\mathsf{I}, \text{TS}_\mathsf{R})$. Upon receiving a message of this form, the responder checks the signature of the message, decrypts the message, and verifies Auth_I. It then adds an entry to its association database for the $\mathsf{I} \rightarrow \mathsf{R}$ association. The inbound policy database is updated to reflect that all traffic matching TS_I should be in the $\mathsf{I} \rightarrow \mathsf{R}$ association.

The responder then fetches its certificate Γ_R. It also generates

$$\text{Auth}_\mathsf{R} = S(SK_a^\mathsf{R}, (\text{Msg2}, n_\mathsf{I}, S(SK_a^\mathsf{R}, \text{ID}_\mathsf{R}))),$$

where ID_R is the identity of the responder. The responder then sends the message

Msg 4 $\mathsf{R} \rightarrow \mathsf{I} : \text{SPI}_\mathsf{I}, \text{SPI}_\mathsf{R}, E^*(SK_\mathsf{I}, (\text{ID}_\mathsf{I}, \Gamma_\mathsf{R}, \text{Auth}_\mathsf{R}))$

After sending the message, the responder adds an entry to the association database for the $\mathsf{R} \rightarrow \mathsf{I}$ association. The outbound policy database is updated to direct all traffic matching the TS_R into the $\mathsf{R} \rightarrow \mathsf{I}$ association.

If I receives a message of this form, it checks the signature, decrypts the message, and verifies Auth_R. The initiator, then adds an entry to its association database for the $\mathsf{I} \rightarrow \mathsf{R}$ association. The outbound policy database is updated to direct all traffic matching the TS_I selector into the $\mathsf{I} \rightarrow \mathsf{R}$ association. An entry to the association database is made for the $\mathsf{R} \rightarrow \mathsf{I}$ association. The inbound policy database is updated to indicate that traffic matching the TS_R selector should travel in the $\mathsf{R} \rightarrow \mathsf{I}$ association.