A Unified Approach to Intra-Domain Security

Craig A. Shue*
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN 37830–6418
Email: shueca@ornl.gov

Andrew J. Kalafut, Minaxi Gupta
Computer Science Department
Indiana University
Bloomington, IN 47405
Email: {akalafut, minaxi}@cs.indiana.edu

***Abstract*—While a variety of mechanisms have been developed for securing individual intra-domain protocols, none address the issue in a holistic manner. We develop a unified framework to secure prominent networking protocols within a single domain. We begin with a secure version of the DHCP protocol, which has the additional feature of providing each host with a certificate. We then leverage these certificates to secure ARP, prevent spoofing within the domain, and secure SSH and VPN connections between the domain and hosts which have previously interacted with it locally. In doing so, we also develop an incrementally deployable public key infrastructure which can later be leveraged to support inter-domain authentication.**

I. Introduction

Intra-domain security is becoming increasingly important. According to a 2007 study [1], 84% of the information security-related incidents could be attributed to current or former employees, often with compromised machines. Unfortunately, the deployed versions of heavily used intra-domain protocols lack strong protection for the network or clients. As a result, misconfigured, malicious, or compromised devices or adversaries masquerading as insiders could launch attacks.

While current approaches to secure the intra-domain network have focused on individual protocols, we instead take a unified approach to all the common protocols, eliminating the need for deployment of multiple security schemes. We first consider the Dynamic Host Configuration Protocol (DHCP) [2] since it is the first protocol an end host invokes to join the network. DHCP lacks built-in authentication, allowing rogue DHCP servers to join the network and misconfigure other machines or malicious DHCP clients to exhaust the resources of a DHCP server. We secure DHCP by allowing both the clients and DHCP servers to authenticate each other and then configure each client with a certificate it can use to authenticate itself to others in the domain.

Next, we consider the Address Resolution Protocol (ARP) [3]. ARP is critical to mapping IP addresses to link layer addresses. Due to a lack of authentication, ARP is vulnerable to cache poisoning attempts from adversaries who act as clients. This poisoning can be used to launch DoS attacks or even sophisticated man-in-the-middle attacks. We leverage the credentials from our modified DHCP to secure ARP and other protocols, yielding a holistic approach to secure intra-domain protocols. In particular, in ARP, we reject any responses which do not contain a valid certificate verifying the address binding.

Other protocols routinely used in the intra-domain setting also suffer from various authentication problems because they lack secure key distribution. The Secure Shell (SSH) protocol [4] asks users to verify public keys while the IPSec protocol [5] requires these keys to be distributed in advance. However, both protocols support the usage of certificates. By providing certificates, our approach makes this support practical.

Our approach does not rely on the presence of any Internet-wide public key infrastructure. Thus, any organization can deploy it independent of any other. At the same time, as our approach is adopted, it can begin to support greater inter-domain security. Our evaluation results show that the cryptographic overheads for the DHCP and ARP protocols can be easily accommodated by existing intra-domain infrastructures, implying that our approach is practically deployable.

II. Security Issues in Intra-domain Protocols

We begin by highlighting the key features and inherent insecurities of commonly used intra-domain protocols. We then discuss prior work that addresses these insecurities. Each of these solutions are specific to the individual protocols, requiring separate deployments for a secure network. Our contribution is a unified system for securing these protocols.

A. DHCP

DHCP provides a framework for passing configuration information to hosts on a TCP/IP network [2] and provides centralized administration, time-sharing of IP addresses, and automatic configuration of new hosts. Unfortunately, DHCP has no built-in authentication mechanism, opening the protocol to attacks. For example, an adversary can join the network as a DHCP server and provide clients with misleading configuration information. A malicious DHCP server could provide hosts with a default router that intercepts traffic. Adversaries could also join the network as clients, spoof MAC addresses, and generate bogus requests to exhaust the pool of

* Craig Shue performed this work while a Ph.D. candidate at Indiana University. Final edits of the submitted manuscript have been authored by a contractor of the U.S. Government under contract DE-AC05-00OR22725. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

IP addresses, preventing legitimate clients from acquiring an IP address.

Two RFCs address the issue of authentication in DHCP. RFC 3118 [6] defines an option which provides authentication and replay detection using shared secrets. This method does not protect the portions of the communication which may be added by a DHCP relay; however, [7] adds this protection. UA-DHCP [8], adds user authentication to DHCP. By requiring the user to supply a user name and password, this system provides access control to the network, but still allows legitimate users access to the network from any machine without requiring MAC address registration. These approaches do not provide credentials for machines in the domain to authenticate each other while our approach would provide cryptographic certificates. Further, since we use asymmetric cryptography, we avoid the administrative overhead of distributing symmetric keys, as required in RFC 3118.

B. ARP

ARP [3] is used to map IP addresses to link layer (commonly MAC) addresses. To determine the MAC address for a given destination, a host consults its local cache. If an entry for the host has been cached, the cached entry is used. Otherwise, the host issues a broadcast query for the destination and uses the MAC address specified in the response it receives. In practice, anyone can respond to these queries and provide false information, poisoning the cache of the querying system. This can be used to DoS a legitimate client or to launch a man-in-the-middle attack to eavesdrop or alter traffic.

One techniques to counter ARP insecurities is *DHCP snooping* [9]. A switch employing this technique monitors DHCP traffic to create white-lists of MAC address and IP bindings, and associate them with individual ports. Subsequently, if a packet arriving on a switch interface does not match the binding, it is discarded. This approach eliminates the possibility of ARP cache poisoning attacks and IP spoofing. S-ARP [10] provides each host with a public/private key pair which they use to sign each ARP message. We use a similar method for securing ARP but avoid the S-ARP requirement for an Authoritative Key Distributor by providing certificates along with the ARP messages. S-ARP also requires hosts to be pre-configured with a valid key pair, while our system allows establishment of credentials upon joining. The Secure Address Resolution Protocol [11] secures ARP by using shared secrets and a central server, which regulates all ARP communication. Hosts periodically communicate their IP and MAC addresses to this server, which answers all ARP requests. TARP [12] operates by having a Ticketing Agent issue signed tickets to each host with the host's IP/MAC mapping, which are sent along with ARP replies. Our system uses a similar approach for ARP, but uses certificates signed by the DHCP server instead of tickets. The use of certificates provides protection against impersonation. SEND, defined in RFC 3971 [13], secures IPv6 neighbor discovery, the IPv6 equivalent of ARP. This is done by adding timestamps, nonces, RSA signatures, and cryptographically generated addresses [14].

C. SSH and IPSec

SSH [4] is an application layer protocol used for secure remote access. It also allows for secure file transfers and port forwarding. Once established, the communication between parties in the SSH protocol is secure; however, there are weaknesses in the initial authentication of the remote server. Typically, clients are asked to verify the public key associated with the server they request. The user generally has two options: 1) obtain the public key fingerprint in an out-of-band fashion or 2) simply take a "leap of faith" and accept the key without verifying it. Upon doing so, the public key is entered into a database and a warning issued if a different key is ever presented for the same host. The leap-of-faith approach provides protection against subsequent man-in-the-middle attacks, but is vulnerable during the initial connection.

The IPSec protocol [5], a popular virtual private network (VPN) protocol, allows hosts to establish cryptographic tunnels to transmit data. These packets can be solely authenticated or authenticated and encrypted. IPSec provides strong confidentiality and authenticity for communications. However, like SSH, IPSec requires another mechanism for the end-hosts to authenticate each other.

III. Overview of our Approach and Threat Model

A. Threat Model

We focus on two categories of attacks. The first are attacks that adversaries can launch by masquerading as legitimate clients or servers. The second are attacks that clients and servers can launch by impersonating other clients or servers. In both of these categories, we ignore attacks that simply aim to congest links with spurious packets and in turn launch DoS attacks. We assume that hosts that simply flood the network with traffic will be quickly identified by an administrator and removed from the network.

B. Overview of our Approach

Our approach assumes that each organization has generated a (public, private) key pair for use in its domain locally and that each server and client in the domain is configured with the public key of the domain. We also assume that the domain uses its private key to sign the public key of each server in the organization in the form of a certificate. We can then leverage these certificates to authenticate the servers in various intra-domain protocols.

We use the DHCP protocol to issue certificates to hosts. Clients begin by using the domain's public key, the certificate from a given DHCP server, and a signature from that server to verify that a given DHCP server is authorized. This process eliminates the attacks introduced by rogue DHCP servers. The DHCP server authenticates the client by requiring credentials, such as a user name and password, before providing it with a viable configuration information. This two-way authentication ensures that only authorized clients are allowed to join the network, thus eliminating attacks from adversaries who exhaust the pool of IP addresses. Before obtaining configuration information securely through DHCP, the client also generates

a (public, private) key pair. Upon verifying the authenticity of the client, the DHCP server issues the client a certificate, which contains the client's public key. Other intra-domain protocols can naturally leverage this certificate.

Figure 1 shows the certificate chain hierarchy within the `example.com` domain. Each server shown has a certificate signed by the private key of the domain while the hosts acquire their certificates from the DHCP server.

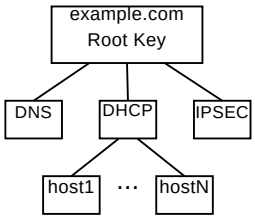

Fig. 1. Example certificate chain hierarchy.

We use traditional X.509 [15] certificates with additional extensions to convey permission within the domain and to provide a strong binding between MAC and IP addresses. Specifically, they have a notion of *access flags*, indicating the rights of the certificate owner. Thus, when verifying the validity of certificates, members of the domain can seamlessly determine whether the certificate issuers were authorized. For example, the certificate for the DHCP server(s) will have a flag that the server is authorized to sign certificates for the hosts configured via DHCP. Client certificates are similar but have different extension options. Finally, the certificates contain an issue date and an expiration date. The issue date is the time when the DHCP server issues the certificate and the expiration is set to when the DHCP lease will expire[1]. With short lease times, DHCP servers can time-share IP addresses without being required to revoke client certificates.

IV. Securing DHCP

Our approach relies on the DHCP server being able to associate a public key with every user on a machine. We first discuss the case of returning users. For these users, the DHCP server already has certification authorizing these public keys for network access. The case of how new users register a (public, private) key pair is discussed in Section IV-B.

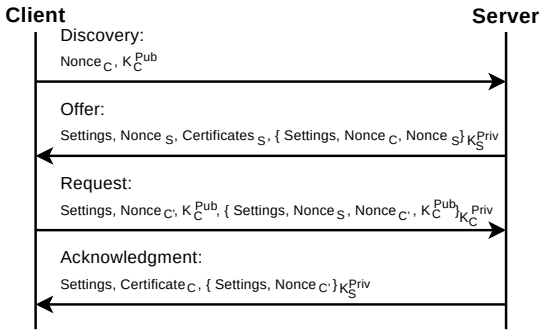

Fig. 2. DHCP secured by our approach

[1]Protocols such as IPSec or SSH do not require MAC or IP addresses in their certificates. A separate, long-term certificate can be used instead.

A. Proposed DHCP Operation

The process of obtaining an IP address and configuration information through the traditional version of DHCP in use today consists of the following four steps. In the first step, the client wishing to join the network issues a broadcast DHCP `Discovery` message on the network. The client seeks an IP address through this message. This message may also indicate the address and other settings the client would prefer to receive. The DHCP server responds to the client in unicast with a configuration `Offer` message which includes the IP address it is offering. This message may include settings such as subnet information, a default gateway, DNS servers, lease length, and address information for the host, among other things. Next, the client broadcasts a `Request` message for the offered configuration if it likes the offer. If the configuration is still available, the server finally responds with an `Acknowledgment` message, at which point the client is configured according to the offered information.

We modify each of the four messages in the DHCP exchange to include cryptographic primitives, as shown in Figure 2. We now describe each of the modified messages.

- **Discovery Message:** We introduce two additional parameters in this message. The first is the public key of the user on a machine, which is used by the DHCP server to retrieve the previous settings for this client. If the client possesses multiple MAC addresses, the MAC address contained in the link layer header of the packet can be used to retrieve the appropriate settings. The second is a nonce value, which helps ensure the freshness of the response from the server.
- **Offer Message:** Upon receiving the `Discovery` message, the DHCP server determines the offer to make. This step is similar to its counterpart in traditional DHCP except that the public key of the client is used to retrieve previous settings instead of simply the MAC address. To prevent rogue DHCP servers from misconfiguring the client, we require that the server provide proof that it is legitimately associated with the domain and that it is not simply replaying an older response. To meet the first requirement, the server includes a full certificate chain to a root of the domain. The client can verify this chain by using the public key of the domain that all clients are configured with. To meet the second requirement, the DHCP server replies with a signature covering the nonce value sent by the client in the DHCP discovery message. The server also includes its own nonce value to ensure the client's liveliness. To prove authenticity, and to prevent modification of the values in transit, the offered settings and nonce value are signed by the DHCP server's private key.
- **Request Message:** Upon receiving a DHCP offer, the client must determine whether to accept the offer. To avoid being misconfigured, it first determines if the certificate chain in the `Offer` message includes a certificate signed by the domain key, which the client also trusts. If so, the client can use the certificate chain to verify each certificate and subsequently the DHCP server. If the client cannot find a trusted root in the certificate chain, the client must reject the offer. Once

the public key for the DHCP server is validated, the client verifies that the signature for the offer is correct. Next, the client evaluates the offer itself, just as in traditional DHCP. If it decides to accept the offer, it broadcasts a DHCP `Request` message for the settings offered. The message also includes the client's public key and a new nonce value that must be echoed by the server. The settings, the server's last nonce value, the client's new nonce, and public key are signed using the client's private key. By signing the server's last nonce value, the client proves the `Request` message is not being replayed.

• **Acknowledgment Message:** In the final step, the DHCP server constructs an `Acknowledgment` message by providing the settings, the nonce value from the client, and a certificate for the host to use with other intra-domain protocols. The settings and nonce value are signed with the server's private key and the signature is included in the message. The client can verify the signature using the certificate information obtained from the DHCP `Offer` message. Further, the client can confirm its issued certificate is valid by using the provided certificate chain and the DHCP server's public key.

B. Bootstrapping New Clients

When a new client first connects to the network, or it connects with a different user than has been seen on it before, it is unrecognized by the network and considered unauthorized. To be bootstrapped, the client must first generate a (public, private) key pair for the current user. It then approaches the DHCP server through the DHCP `Discovery` message as before. When the DHCP server fails to find an entry for this client's public key, the DHCP server provides the client with settings that isolate it from the network and redirects all requests to an organization authentication service, such as RADIUS [16] or a captive portal, which is a secure Web authentication page. When the client authenticates, it provides the user's public key and the client's MAC address to the authentication system, which consults local policy[2], and then communicates the key and MAC address to the DHCP server. This separation of functionality allows DHCP to focus on certifying settings while being compatible with multiple user authentication systems. Once the DHCP server receives the information from the authentication system, it knows that the client is providing a valid public key for some user on the system with the given MAC address. This step ensures that the DHCP server knows the public keys for each authorized user, every valid user in an organization is allowed to acquire only one key per MAC address, and each user is limited to a low number of public keys concurrently registered, preventing the user from spoofing a large number of MAC addresses to exhaust the pool of IP addresses from the DHCP servers.

Notice that our approach does not attempt to prevent MAC address spoofing on a small scale. Further, it does not prevent a user from being associated with several MAC addresses, which may be useful if a user has multiple computers or network interfaces (e.g. wired or wireless).

[2]For example, policy may limit a user to a fixed number of public keys.

C. Discussion of Security

In the secure DHCP protocol, we must ensure the legitimacy of clients and servers engaging in the protocol. We must also ensure that no adversary is able to disrupt correct protocol operation. Further, no entity, including adversaries or legitimate clients, should be able to acquire any more than one IP address per MAC address, subject to a maximum allowable number of MAC addresses. These requirements can be translated into the following five properties:

1) the DHCP server is trustworthy
2) the offered settings are not manipulated by an attacker
3) messages are not replayed
4) each client uses only one key pair at a time
5) the client holds the private key in the key pair

Our protocol design preserves the above properties. 1) The certificate chain ensures the authenticity of the DHCP server, providing the first property. If a DHCP server provides a valid certificate issued by a party trusted by the client, in this case the domain key, the client can be assured that the DHCP server is valid if the digital signature is correct. 2) The DHCP server signs the settings it offers, ensuring that the offered settings are not manipulated by an attacker, guaranteeing the second property. 3) We use signed nonces on both the client and server side to provide the third property, that messages are not replayed. 4) The use of a RADIUS server or captive portal ensures that each user possesses exactly one valid (public, private) key pair per client at any point. This ensures the fourth property, which require that the clients be able to secure only one IP address per allowable MAC address. 5) Finally, since the client must issue a signature to demonstrate it holds its own private key, which guarantees the fifth property.

In our protocol, the DHCP server must perform a public key operation before the client has proven its authenticity. This can lead to DoS attacks targeting the DHCP server's processor resources. Instead, the DHCP server can require the client to reply to a nonce value proving its liveness before generating a signature. This extra round-trip could optionally be required only when the server is under heavy processor load.

V. Securing Other Intra-domain Protocols

In designing our intra-domain security approach, we leverage the DHCP server as the gate keeper for the network. It distributes certificates to hosts in a secure and verifiable manner and thus provides them with a mechanism to prove their authenticity in later communications. We now discuss how this impacts other protocols.

A. Securing ARP

We secure ARP by adding operations to the protocol. Under our scheme, an end host transmits the regular ARP request as is done today. However, when replying to an ARP request, it must include the certificate it obtained from the DHCP server showing the IP and MAC binding. The requester can then verify the certificate to confirm that the response given by the responder is accurate. Clients would reject ARP responses that lack valid certificates. This simple extension eliminates both

ARP cache poisoning attempts. We further note that certificate verification is only required on the order of once every 5 minutes or so, when ARP cache entries expire. We evaluate these overheads in detail in Section VI.

Our modifications to ARP create a secure association between a given MAC and IP address, preventing a host from spoofing only its MAC or its IP address. However, the host could spoof both the MAC and IP address of another valid host. This can be prevented in two ways: by proving authenticity in each packet or leveraging DHCP snooping. To prove authenticity at connection establishment, a client may begin a Diffie-Hellman (DH) exchange and sign its values. The server would then sign a message with its own DH value. In subsequent messages, the client and server may simply use a nonce and the secret key derived from the DH exchange to construct a message authentication code and embed this code in the body of each message, allowing the other client to verify its authenticity. To avoid a DoS attack, the server may force the client to respond to a nonce before verifying the client's signature and beginning the DH exchange.

While effective, providing authenticity of each message incurs a modest additional overhead at the end-hosts. Instead, DHCP snooping by the switching infrastructure can leverage network topology to avoid requiring per-packet verification. Some switches today employ DHCP snooping to prevent ARP cache poisoning and IP spoofing. Essentially, such switches monitor DHCP traffic to create allowable MAC address and IP bindings, and associate them with individual switch ports. When a packet arriving on an interface does not match the binding, the packet is discarded. With our approach, switches can employ stronger DHCP snooping protection.

B. Securing SSH, TLS/SSL, and IPSec

Protocols such as SSH, TLS/SSL, and IPSec can directly use the certificates generated by the DHCP server in our system to prove the public key they provide is authentic.

VI. Implementation and Evaluation

We implemented the modified ARP and DHCP protocols and then experimentally analyzed their performance based on the usage of these protocols on a medium and large network. We focused on these two protocols because we introduce new overheads to them. The rest of the intra-domain protocols we discussed can already incorporate the certificates we provide.

All of the performance trials were conducted on a machine with a Pentium IV 1.8 GHz processor with 512MBytes RAM. To measure the timings, we use the `RDTSC` instruction, yielding nanosecond timing resolution. We used the Botan cryptographic library for C++ [17] in our messages.

A. DHCP

To evaluate our changes to the DHCP protocol, we implemented the entire protocol, adding DHCP options for each of the new fields required in our version, and timed the cryptographic operations. We implemented seven new option types, allowing the specification of a client public key, client and server nonce values, a signature of the DHCP message, certificates for the DHCP client and server, and supporting certificates required to verify the client and server certificates.

TABLE I
Cryptographic operations in our DHCP protocol

Operation	DHCP Message	Machine	Required Time (ms)
Generate Client Key Pair	N/A	Client	145.003
Create Nonce	All but Ack.	Both	0.032
Create Signature	Request	Client	9.428
Verify Message Signature	Request	Server	16.163
Create Signature	Offer	Server	15.618
Verify DHCP Server Certificate	Offer	Client	14.412
Verify Message Signature	Offer	Client	0.900
Create Certificate	Ack.	Server	71.038
Create Signature	Ack.	Server	15.641
Verify Client Certificate	Ack.	Client	19.823

We generated an RSA key pair and root certificate for the `example.com` domain and another for the `dhcp.example.com` DHCP server. We signed the DHCP server certificate with the domain root key. For each DHCP client, we generated a DSA key pair. In the modified DHCP protocol, the DHCP client provides the server with its DSA public key. The DHCP server generates a certificate for the client from this public key, and signs the certificate with its RSA private key. We measured the cryptographic overheads and the message size overheads for both entities.

In Table I, we list the cryptographic operations for the modified DHCP protocol with the machine performing the operation and the associated DHCP message. The most significant overhead is the generation of a DSA key pair on the client; this overhead is not required for each DHCP execution and can be amortized. The next most significant overhead, the creation of the client's certificate, is executed in the last message generated by the server after the client has already proved its liveness and authenticity, providing some protection against spoofed DoS attack attempts.

To determine if these overheads are acceptable, we examine two network deployments: a smaller, largely static network with about 570 hosts and a larger, dynamic network with about $111,500$ hosts registered. The smaller network has a single DHCP server with 100 IP addresses in its pool with 55 in use at the time measured. The server used a lease time of 24 hours. During a 24 hour period, the server received $1,027$ DHCP messages, an average of roughly 1 request every 1.4 minutes. The cryptographic overheads are unlikely to be detrimental to this DHCP server's operation. The larger network has two DHCP servers with two different lease times: 8 hours for wired connections and 2 hours for wireless connections. During one day of operation, these servers received $271,324$ new lease requests and $215,640$ renewal requests. Accordingly, the two DHCP servers received an average of $20,290$ requests per hour. If this load were divided evenly, the servers processed about 2.82 requests per second. The cryptographic overheads at the server on our test machine were about 119ms per client request. Considering only these overheads, our modest test server could process 8.44 sequential requests per second even without exploiting any parallelism. Accordingly, we believe our proposed solution is feasible for both these networks.

We next examine the size of each message in the modified DHCP protocol. The DHCP discovery and request messages easily fit within a single Ethernet frame, requiring 955 and $1,025$ bytes respectively. The DHCP offer and acknowledgments, which require $2,313$ and $1,866$ bytes respectively, exceed the standard $1,500$ byte MTU for Ethernet. Some networks can easily accommodate these messages while other networks will split the packets into two fragments and send them together with little overhead.

B. ARP

The operations required in the modified ARP protocol are a proper subset of the functionality in the modified DHCP protocol. Hosts issuing ARP requests have no extra overheads when creating the request. However, they must verify the certificates in any requests they receive, an operation which takes approximately 19.8ms on our test machine. Hosts responding to ARP requests must provide their own certificates in addition to the ARP reply header, but this operation does not introduce any cryptographic overhead. These overheads seems acceptable for hosts, which use ARP relatively infrequently.

These overheads become more acute for routers and Ethernet switches. We again turn to our example network deployments to determine the feasibility of processing these ARP messages. The smaller network is serviced by four Ethernet switches with a link to an external router. The ARP cache expires every 5 minutes on these switches. We monitored the ARP churn on one of these switches for a five minute window and found that 50 cached entries were removed while 42 were added. The switches only challenge new cache entries in order to ensure the mappings are valid. Accordingly, these switches would need to generate an ARP request and verify a certificate once every 7.14 seconds. This verification can be easily accommodated, requiring about 19.8ms on our test machine. Our larger example network has 5 routers, with the busiest router peaking at about $21,000$ ARP cache entries. At peak times, a few thousand ARP cache entries are added per hour. We conservatively estimate a peak addition rate of half of the observed cache size, or $10,500$ entries per hour, or approximately 3 per second. Excluding non-cryptographic overheads, the router could issue and verify 50.45 requests per second, easily accommodating this overhead. Routers must also reply to ARP requests from hosts; however, these replies do not incur a cryptographic overhead. We conclude the computation overheads to be acceptable for these networks.

We next examined the packet size of ARP requests and replies. A regular host ARP request is just 28 bytes, the size of the ARP header. ARP replies with a certificate require $1,395$ bytes. Both messages fit inside a regular Ethernet frame, eliminating the need for additional messages.

VII. Discussion

In this work, we introduced and evaluated a unified framework for authenticating and authorizing machines within a domain and found it to be feasible. Here, we highlight additional important issues in realizing the proposed architecture.

Obtaining a Key for a Domain: We only discussed the case where domain root keys were pre-loaded at the users. An alternative could be to ask users to enter or verify a public key upon connection. This is similar to modern wireless security protocols. Also, we could leverage certificate authorities or secure, DNSSEC-protected, DNS records.

Certificate Revocation: Organizations must be able to revoke compromised certificates to avoid misuse. Prevalent solutions to accomplish this goal, namely certificate revocation lists (CRLs) [18] and the Online Certificate Status Protocol (OCSP) [19] can be used seamlessly in our infrastructure.

Future Directions: Multi-user systems and low-power devices pose unique challenges for the architecture we propose. These subjects are both worthy of future exploration.

References

[1] PricewaterhouseCoopers, "The global state of information security," 2007, http://www.pwc.com/extweb/pwcpublications.nsf/docid/114E0DE67DE6965385257341005AED7B/$FILE/PwC_GISS2007.pdf.

[2] R. Droms, "Dynamic host configuration protocol," IETF RFC 2131, Mar. 1997.

[3] D. Plummer, "An Ethernet address resolution protocol," IETF RFC 826, Nov. 1982.

[4] T. Ylonen and C. Lonvick, "The secure shell (SSH) protocol architecture," IETF RFC 4251, Jan. 2006.

[5] S. Kent and R. Atkinson, "Security architecture for the Internet protocol," RFC 2401 (Proposed Standard), Internet Engineering Task Force, Nov. 1998, updated by RFC 3168.

[6] D. Droms and W. Arbaugh, "Authentication for DHCP messages," IETF RFC 3118, June 2001.

[7] M. Stapp and T. Lemon, "The authentication suboption for the dynamic host configuration protocol (DHCP) relay agent option," IETF RFC 4030, Mar. 2005.

[8] T. Komori and T. Saito, "The secure DHCP system with user authentication," in *IEEE International Conference on Local Computer Networks (LCN)*, 2002.

[9] Cisco Systems, "Cisco DCNM security configuration guide, release 4.0 - configuring DHCP snooping," http://www.cisco.com/en/US/docs/switches/datacenter/sw/4_0/dcnm/security/configuration/guide/sec_dhcpsnoop.html.

[10] D. Bruschi, A. Ornaghi, and E. Rosti, "S-ARP: a secure address resolution protocol," in *Annual Computer Security Applications Conference (ACSAC)*, 2003.

[11] M. Gouta and C. Huang, "A secure address resolution protocol," *Computer Networks*, pp. 57–71, Jan. 2003.

[12] W. Lootah, W. Enck, and P. McDaniel, "TARP: Ticket-based address resolution protocol," in *Annual Computer Security Applications Conference (ACSAC)*, 2005.

[13] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "Secure neighbor discovery (SEND)," IETF RFC 3971, Mar. 2005.

[14] T. Aura, "Cryptographically generated addresses (CGA)," IETF RFC 3972, Mar. 2005.

[15] R. Housley, W. Ford, and D. Solo, "Internet X.509 public key infrastructure certificate and CRL profile," IETF RFC 2459, Jan. 1999.

[16] C. Rignet, S. Willens, A. Rubens, and W. Simpson, "Remote authentication dial in user service (RADIUS)," IETF RFC 2865, June 2000.

[17] J. Lloyd, "Botan," http://botan.randombit.net/.

[18] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," IETF RFC 5280, May 2008.

[19] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "Internet X.509 public key infrastructure online certificate status protocol - OCSP," IETF RFC 2560, June 1999.