SANDIA REPORT
SAND2006-7078
Unlimited Release
Printed November 2006

Final Report for the Network Authentication Investigation and Pilot

Edward L. Witzke, John M. Eldridge, Marc M. Miller, Dallas J. Wiener,
and Nathan Dautenhahn

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2006-7078
Unlimited Release
Printed November 2006

Final Report for the Network Authentication Investigation and Pilot

Edward L. Witzke, John M. Eldridge, Marc M. Miller, Dallas J. Wiener,
Nathan Dautenhahn
Advanced Networking Integration Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0806

Abstract

New network based authentication mechanisms are beginning to be implemented in industry. This project investigated different authentication technologies to see if and how Sandia might benefit from them. It also investigated how these mechanisms can integrate with the Sandia Two-Factor Authentication Project. The results of these investigations and a network authentication path forward strategy are documented in this report.

Acknowledgements

The authors wish to thank the following people for their contributions to various aspects of this project:
Diana Eichert
Steve Gossage
Jim Schutt
Dan Wachdorf

Contents

List of Figures

List of Tables

1 Introduction

The Mobile Computing Initiative is to provide the necessary connectivity to facilitate a new culture of mobile employees, including higher user productivity, enhanced quality of work, increased flexibility, and greater employee satisfaction. One piece of that initiative is this project, the Network Authentication Investigation.

New network based authentication mechanisms are beginning to be implemented in industry. Network based authentication protocols, such as 802.1x, provide increased network security by authenticating the machine and/or user prior to providing access to any network resources. Network based authentication may also provide some additional capabilities, such as remediating and patching unpatched/un-updated computer systems prior to them accessing internal Sandia networks.

This project investigated different authentication technologies (especially 802.1x port authentication and network node validation and remediation solutions) to see if and how Sandia might benefit from them. It also investigated how these mechanisms can integrate with the Sandia Two-Factor Authentication Project. The results of these investigations and a network authentication path forward strategy are documented in this report.

2 Network Authentication Technology

2.1 Purpose and Overview

This section is intended to describe the current state of network authentication technology. This section surveys the existing technologies, their advantages, their disadvantages, and describes equipment suitable for use with some of these technologies.

2.2 Types of Network Authentication

2.2.1 802.1x

The 802.1x protocol uses EAPOL (Extensible Authentication Protocol Over LAN) to authenticate users to a network on a port by port basis. It is possible to implement 802.1x via wireless or wired network configurations. This overview covers the wired side implementations.

There are three major components when using 802.1x authentication. They are the supplicant, the authenticator, and the authentication server. The supplicant is the client attempting to gain access to the network, the authenticator is the 802.1x enabled switch, and the authentication server is a backend device such as a RADIUS server that does the actual authentication.

In an 802.1x enabled network, switch ports are configured to be disabled until a successful authentication occurs. When in the disabled mode, the port is considered to be in an uncontrolled state. During this phase the switch will only allow EAP (Extensible Authentication Protocol) messages to be sent over the uncontrolled port. Upon successful authentication, the authentication server sends an access accept message to the switch, which then opens the specific port to which the client is connected, for all protocols and places it in a controlled state.

When a client first connects to the port, it sends an EAP request to the switch, which then forwards the request to the authentication server. Upon receiving the request the authentication server sends a response with a challenge in it for the client to send its credentials. The client then sends its credentials to the server, which will accept or reject the client.

2.2.2 Static Password

The purpose of a password is to prevent unauthorized people from accessing files, systems, or networks. It also authenticates authorized users, since only that user is supposed to know the corresponding password. In her 1982 book [4], Dorothy Denning describes login protocols using passwords. These have not changed much over the years.

In static password systems, a password is assigned to each user and is changed either periodically or aperiodically by the user, depending on local policies. These passwords may be stored in memory or on disk in cleartext or encrypted form. Another storage method is to encode the password and then using the encoded password to encrypt a known block of data (such as a block of zeros) [7]. At login time, this is compared to an encryption performed on the same known block using the encoded form of what the user entered for his password, as the encryption key. If the results match, the user is allowed to proceed.

2.2.3 One Time Passwords

A one time password (or single use password) is a password that changes with each use. Each time a user correctly presents their password to the authentication system, it changes [6]. In this way, even if a password is observed by "shoulder surfing" or network sniffing, it's no longer of use because the authentication system has cycled to the next password in the list or algorithm.

Some one time password systems are single factor systems using prepopulated lists of passwords. Most one time password systems operate on the principle of two-factor authentication, requiring something you have, such as a token card, and something you know, for example a personal identification number (PIN) [3].

2.2.4 Network Node Validation

Last year vendors began advertising new and upcoming solutions that aim to provide enterprises with better network access control. These new solutions, commonly referred to as network node validation (NNV) or network access control (NAC), provide network access control decisions based upon the credentials of the computer and/or user of the computer and the health, or security posture, of the computer. In addition to the individual vendor products, the Trusted Network Connect (TNC) working group of the Trusted Computer Group (TCG) has recently created an open, standards-based architecture for endpoint security and access control. This architecture should promote interoperability between network infrastructure vendors and end-point security vendors.

Another major component of the NNV is the ability to remediate or quarantine a virus or worm infected client prior to allowing access to the internal network. This allows enterprises to ensure that all machines are patched, have current signature files for anti-virus and host-based intrusion detection, and have anti-virus, firewall, and intrusion detection running. Infected or unpatched machines would be cleaned or patched prior to allowing network access to the internal network.

Two major products in this category are the Cisco Network Admission Control (NAC) framework and the Symantec-owned Sygate Universal NAC. Both products operate similarly, and in addition to the main functionality of NNV, these products also permit dynamic VLAN and ACL assignment based upon the credentials of the user or the posture of the computer. Figures 1 and 2 detail the step-by-step operation of both the

Sygate NAC and the Cisco NAC, respectfully, in a wired campus environment. These solutions also allow the same functionality to be used for SSL VPN and IPSEC VPN connections.

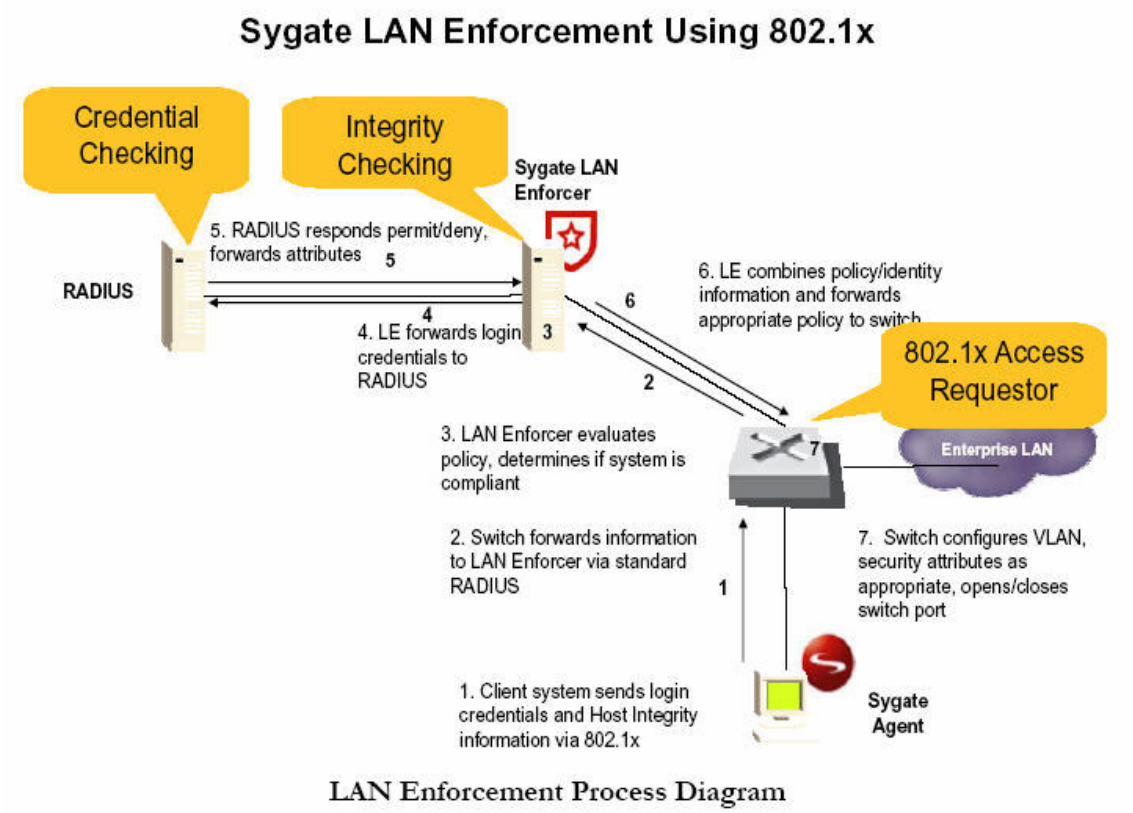

Figure 1. Symantec-owned Sygate Universal NAC flow diagram [11].

An important thing to note is that both solutions require system operators to install an agent on the client machine. This agent communicates with software plug-ins that report the status of anti-virus, firewall, or host-based IDS software. The agent then passes this information on to the policy server that makes a decision on client compliance.

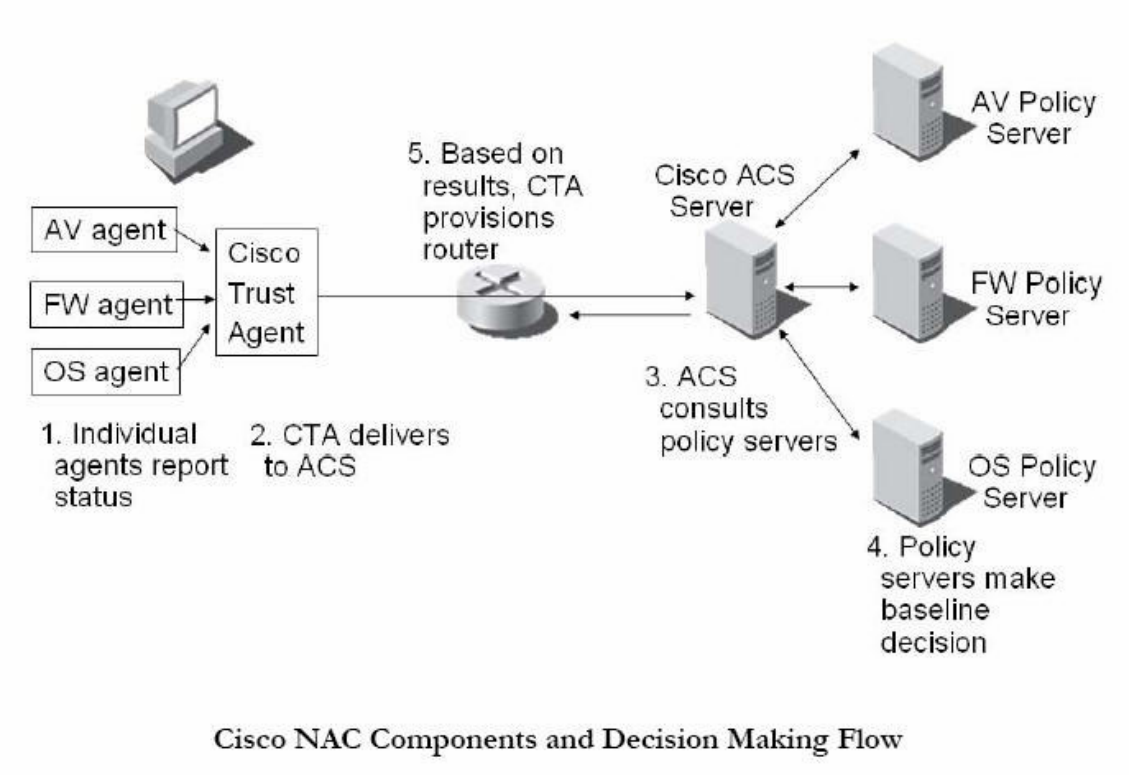

Figure 2. Cisco NAC flow diagram [11].

2.3 Advantages and Disadvantages of Network Authentication Types

2.3.1 802.1x

The advantages and disadvantages within the 802.1x framework depend greatly on the EAP type being used. Some EAP types are more robust or more vulnerable than others. Some, like TLS provide a high degree of security with two factor authentication.

The major advantage of 802.1x authentication is the fact that only EAP traffic is allowed over switch ports until users transact a valid authentication. This is important in locations where physical security is not completely controlled, since it prevents a perpetrator from connecting to the network by just having physical access.

Another advantage that 802.1x offers, while using a secure EAP type like EAP-TLS, is that the client must be connected to a trusted authentication server. This helps prevent a "rogue authentication server" from authenticating users and controlling their connection. The requirement is that both ends of the solution must authenticate to each other. Similarly, another advantage is that the switch can only communicate with predefined authentication servers, providing more security to the backend implementation.

The 802.1x standard offers several different types of EAP and backend authentication types, such as RADIUS. This allows for flexible implementation when installing 802.1x

in an existing network. By using EAP-TLS with smart cards, a two factor authentication scheme can be implemented.

A potential perpetrator could possibly set up a hub in between a true client and a switch. They could then watch the traffic and possibly spoof a MAC address and get into the network.

The 802.1x standard details a fairly new network access methodology. Devices such as printers do not fully conform to this methodology yet. This could be a potential hazard because a switch port with a printer connected would then have to be left open (a non-802.1x port), requiring it to be protected by other means, such as with physical security measures.

2.3.2 Static Password

"Passwords should be easy to remember but hard to figure out, guess or crack.[1]" Many static passwords are poorly chosen, making them susceptible to dictionary attacks. A big problem with static passwords is that they may be stolen or copied without the user being aware of the compromise. A static password may be copied for future use by "sniffing" the network connection, if the password is sent unencrypted. Even if it is sent encrypted, a replay attack may be useful, capturing the password off the network and replaying it at a future time to gain access.

Well chosen static passwords may still be suitable for local authentication within a system. For authentication to a network or to another system connected to a network, extreme care must be exercised when developing the authentication system, so to thwart replay attacks. These could be mitigated by encrypting the password differently each time it is sent, but this starts to resemble a one time password system.

2.3.3 One Time Passwords

Advantages of one time password systems are that users cannot choose weak passwords and, if using two-factor authentication, only need to remember a PIN rather than a strong, potentially complicated, password. Also, one time passwords are invalidated after their first use, therefore sniffed passwords are useless as soon as they are acquired from the network.

The disadvantages of one time password systems based on two-factor authentication are that users need to physically have their token cards with them to authenticate, and that it takes longer to authenticate using one of these systems. Two-factor systems may also require an additional server and could be expensive among large populations of users (by having to supply and manage many token cards). Disadvantages of single factor one time password systems include distribution, protection, and synchronization of the prepopulated password lists.

---

[1] Essential System Administration [7], by Æleen Frisch, pg. 149.

2.3.4 Network Node Validation

Network Node Validation products and solutions promise to enable enterprises to permit or deny network access based upon the identity of the user and/or machine as well as the security posture of the machine. In addition these solutions permit quarantine and remediation capabilities for machines that don't have current patch levels, have outdated anti-virus signature files, or have been infected with a virus or worm. Detailed lab testing and usage experience will determine whether these solutions can feasibly be deployed in an enterprise environment.

The disadvantage to these solutions is that they all require software agents and plug-ins to be installed on every host machine in order to provide the full functionality. They also require interaction and cooperation between multiple computing areas such as desktop software, networking, security, and server administration.

A specific disadvantage of the Cisco NAC solution is that it currently only works with Cisco hardware. Cisco has apparently released the information on its NAC framework so that other vendors can use this as well. Time will tell whether other vendors intend to implement the Cisco NAC solution in their network infrastructure hardware.

2.4 Authentication Services

2.4.1 RADIUS

A RADIUS server is typically used in 802.1x authentication as the authentication server. There are several different RADIUS servers available for production. Among them are Radiator, FreeRadius, and several others. Sandia has a license for the Radiator product. Radiator provides modules for every type of EAP and several authentication types. Radiator works by defining an EAP type to communicate with the client. Then, once a connection is established, Radiator will authenticate using several different options for validating the user's credentials. Among these is using a smart card with TLS, an NT domain, querying an LDAP database, local text database, and numerous other types.

2.4.2 TACACS+

TACACS+, a Cisco proprietary protocol, offers centralized authentication services for a network [3]. Sandia's Cisco network hardware uses TACACS+ to authenticate logins. The Cisco Secure ACS (Access Control Server) is the application used to provide TACACS+ authentication.

2.4.3 Kerberos

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. It allows users and services to authenticate themselves to each other. That is, it allows them to

unequivocally demonstrate their identity to each other. Kerberos was designed to eliminate the need to demonstrate possession of private or secret information (the password) by divulging the information itself.

The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network. Kerberos is based on the key distribution model developed by Needham and Schroeder [13]. A key is used to encrypt and decrypt short messages, and is itself typically a short sequence of bytes. Keys provide the basis for the authentication in Kerberos.

Roughly speaking, an encryption routine takes an encryption key and a plaintext message, and returns ciphertext. Conversely, the decryption routine takes a decryption key and the ciphertext, and returns (if decryption is successful) the original plaintext. In Kerberos, at the present time, the encryption key and the decryption key are identical. This is the hallmark of conventional cryptography, in which the keys are either identical or at least easily derivable from one another. In fact, in many realizations, either key can be used for encryption, and the other key for decryption. In contrast, in public key cryptography, there are two keys, one for encryption, one for decryption, which are not derivable from one another.

Kerberos is typically used when a user on a network is attempting to make use of a network service, and the service wants assurance that the user is who he says he is. To that end, the user presents a ticket that is issued by the Kerberos authentication server (AS). The service then examines the ticket to verify the identity of the user. If all checks out, then the user is accepted. Therefore, this ticket must contain information linking it explicitly to the user. The ticket must demonstrate that the bearer knows something only its intended user would know, such as a password. Furthermore, there must be safeguards against an attacker stealing the ticket, and using it later.

Both the user and the service are required to have keys registered with the AS. The user's key is derived from a password that he chooses; the service key is a randomly selected key.

For the purposes of this explanation, let us imagine that messages are written on paper (instead of being electronic), and are ‘encrypted’ by being locked in a strongbox by means of a key. In this ‘box world,’ principals are initialized by making a physical key and registering a copy of the key with the AS.

1. First the user sends a message to the AS: “Alice the User, would like to talk to Bob the Server.”
2. When the AS receives this message, it makes up two copies of a brand new key. This is called the session key that is used in the direct exchange between user and service.
3. The AS puts one of the session keys in Box 1, along with a piece of paper with the name “Bob the Server” written on it. It locks this box with the user's key. The AS includes the paper since, if Box 1 only contained the session key, then the user

wouldn't be able to tell whether the response came back from the AS, or whether the decryption was successful. By putting in "Bob the Server," the user will be able to verify both that the box comes from the AS, and that the decryption was successful.

4. The AS puts the other session key in a Box 2, along with a piece of paper with the name "Alice the User" written on it. It locks this box with the service's key.
5. The AS returns both boxes to the user.
6. The user unlocks Box 1 with his key, extracting the session key and the paper with "Bob the Server" written on it.
7. The user can't open Box 2 since it's locked with the service's key. Instead, he puts a piece of paper with the current time written on it in Box 3, and locks it with the session key. He then hands both boxes (Box 2 and Box 3) to the service.
8. The service opens Box 2 with its own key, extracting the session key and the paper with "Alice the User" written on it. It then opens Box 3 with the session key to extract the piece of paper with the current time on it. These items demonstrate the identity of the user.

The timestamp is put in Box 3 to prevent someone else from copying Box 2 (remember, these are simply electronic messages) and using it to impersonate the user at a later time. Because clocks don't always work in perfect synchrony, a small amount of leeway (about five minutes is typical) is given between the timestamp and the current time. In addition, the service maintains a list of recently sent authenticators, to make sure that they aren't resent in quick order.

To open Box 2, the service uses a randomly generated key that is stored in a special file call a service key file. This file is assumed to be secure, so that no one can copy the file and impersonate the service to a legitimate user. In Kerberos terminology, Box 2 is called the ticket, and Box 3 is called the authenticator. The authenticator typically contains more information than what is listed here. Some of this added information arises from the fact that this is an electronic message (for example, there is a checksum). There may also be an encryption key in the authenticator to provide for privacy in future communications between the user and the service.

There is a subtle problem with the above exchange. It is used every time a user wants to contact a service. But then he has to enter in a password to unlock Box 1 with the key each time. The obvious way around this is to cache the key derived from the password. However, caching the key is dangerous. With a copy of this key, an attacker could impersonate the user at any time until the password is next changed.

Kerberos resolves this issue by introducing a new agent, called the Ticket Granting Server (TGS). The TGS is logically distinct from the AS, although they may reside on the same physical machine. They are often referred to collectively as the KDC -- the Key Distribution Center, from Needham and Schroeder [13]. The function of the TGS is as follows. Before accessing any regular service, the user requests a ticket to contact the TGS, just as if it were any other service. This ticket is called the ticket granting ticket (TGT).

After receiving the TGT, any time that the user wishes to contact a service, he requests a ticket not from the AS, but from the TGS. Furthermore, the reply is encrypted not with the user's secret key, but with the session key that the AS provided for use with the TGS. Inside that reply is the new session key for use with the regular service. The rest of the exchange now continues as described above.

The advantage this provides is that while passwords usually remain valid for months at a time, the TGT is good only for a fairly short period, typically eight hours. Afterwards, the TGT is not usable by anyone, including the user or any attacker. This TGT, as well as any tickets that you obtain using it, are stored in the credentials cache.

2.4.4 LDAP

The Lightweight Directory Access Protocol (LDAP) is a networking protocol for querying and modifying directory services running over TCP/IP. An LDAP directory follows the X.500 model: It is a tree of entries, each of which consists of a set of named attributes with values. An LDAP directory typically reflects political, geographic, and/or organizational boundaries. LDAP deployments today tend to use Domain Name System (DNS) names for structuring the top levels of the hierarchy. Further below might appear entries representing people, organizational units, printers, documents, or anything else.

LDAP is not limited to contact information, or even information about people. LDAP can be used to look up encryption certificates, pointers to printers and other services on a network, and provide ‘single sign-on’ where one password for a user is shared between many services. LDAP is appropriate for any kind of directory-like information, where fast lookups and less-frequent updates are the norm. In short LDAP provides the frame work for establishing a distributed database and allows for the distributed management and update of data.

LDAP is not specifically designed as an authentication service, but rather is a distributed database. In some instances it may useful to store information that could be useful in determining authentication characteristics and options for client/server inter-operations.

2.4.5 NT Domain

Microsoft Windows NT Domains are logical groupings of workstation and user accounts with central administrative control, security, and authorization. While often still referred to as “NT Domain”, it has evolved into “Active Directory”, which is basically the same concept, but built on a different structure with significantly advanced features.

NT Domain authentication utilizes a database of usernames and passwords for authenticating users. The Active Directory implementation uses an LDAP-compatible (but not LDAP-compliant) directory database.

Domain controllers use the information in the directory database to authenticate users logging on to domain accounts.

Non-Windows entities can use an NT Domain for authenticating users, provided the entity speaks the correct protocols. For example, a RADIUS server can utilize an NT Domain or Active Directory to authenticate a user that is attempting access from a dial-up, VPN, or wireless system.

2.5 Equipment Survey

2.5.1 Switches and Operating Software

The 802.1x standard is now supported by the current IOS version of both Cisco and Foundry. This covers all of the switches including the entire Foundry FastIron Edge Series, and Cisco switches 2900 and more advanced.

2.5.2 Operating Systems and Clients

The following chart (Table 1) lists operating systems and clients that provide 802.1x support. It also lists the supported EAP types. This chart can be found at http://www.open.com.au/radiator/technical.html#wireless.

Table 1. Client support for 802.1x.

OS	Client	EAP-Types Supported
Linux, Open BSD	Xsupplicant	MD5, TLS, TTLS, (PAP, CHAP, MSCHAP, MSCHAPV2), PEAP (MSCHAPV2), EAP-SIM
Linux	MDC Aegis	MD5, TLS, TTLS, LEAP, PEAP (MSCHAPV2)
Windows	Window XP	TLS, PEAP (MSCHAPV2, TLS)
	Windows 2000	TLS, PEAP (MSCHAPV2, TLS)
	MDC Aegis	MD5, TLS, TTLS (PAP, CHAP, MSCHAP, MSCHAPV2, EAP-MD5), PEAP (MSCHAPV2, EAP-TLS, EAP-Generic-Token), LEAP, EAP-SIM (with Radiator add-on EAP-SIM support package)
	Alfa+Ariss Secure W2	TTLS-PAP
	Cisco	LEAP, PEAP, EAP-SIM (with Radiator add-on EAP-SIM support package)
	Odyssey Client	MD5, TTLS (PAP, CHAP, MSCHAP, MSCHAPV2), EAP-Generic-Token, TLS, PEAP (MSCHAPV2, EAP-Generic-Token), LEAP
	Boingo	TLS, PEAP (MSCHAPV2, TLS)
Pocket PC	PocketPC 2003 Native	TLS, PEAP (MSCHAPV2, TLS)
	Alfa+Ariss Secure W2	TTLS-PAP
Mac OSX	Xsupplicant	MD5, TLS, TTLS (PAP, CHAP, MSCHAP, MSCHAPV2), PEAP (MSCHAPV2), EAP-SIM
	Panther Native	MD5, TLS, TTLS (PAP, CHAP, MSCHAP, MSCHAPV2), PEAP (MSCHAPV2), LEAP
	MDC Aegis	MD5, TLS, TTLS, LEAP, PEAP

2.5.3 EAP Types

There are several different types of authentication.

- LEAP
  - Proprietary implementation by Cisco
  - Provides mutual authentication
  - Provides frequent authentication and on the wireless side upon every new authentication a client receives a new WEP key

- EAP-TLS
    - Provides mutual authentication
    - Uses PKI infrastructure
    - Requires client side and server side certificates
    - By using a smart card, TLS allows for two factor authentication, both the client and the smart card are authenticated
- EAP-MD5
    - User name and password only for authentication
    - Turns the password into a hash
    - Vulnerable to dictionary attacks
- EAP-TTLS
    - Two stage authentication
    - In first stage the RADIUS server is authenticated to the client creating a TLS tunnel
    - In the second stage the user's credentials are passed to the RADIUS server as attribute value pairs
- PEAP
    - Works much in the same way as TTLS
    - Uses server-side certificates only
    - In addition to the first session created in a TLS tunnel, it performs authentication in a second tunnel

Each of these offers a different scheme with advantages and disadvantages. The most secure method is TLS because of its mutual authentication and the fact that it allows for two factor authentication. The downside to TLS is that there is a lot of overhead when configuring devices with certificates.

3 Network Node Validation and Remediation

Members of this project examined, to varying degrees, network node validation and remediation product offerings from Cisco, ConSentry, Symantec-Sygate, and Vernier. We acquired loaner equipment from ConSentry for more detailed evaluation.

The equipment from ConSentry consisted of a management unit (using a Windows 2003 Server operating system) designed to manage multiple ConSentry LANShield switches and LANShield controllers, and a LANShield CS2400 controller. The CS2400 has 10 pairs of ports supporting small form factor GBICs. The LANShield controller would normally sit between a distribution layer switch and an access layer switch. In our lab, the CS2400 was wired up such that one port of the pair (uplink) was connected to a Foundry FastIron 2402 (representing the distribution layer switch) and the other port of the pair (downlink) was connected to a 3Com SuperStack 3 Switch 4400 (representing the access layer or edge switch), as shown in Figure 3. A laptop computer, representing a portable client, was plugged into the 3Com switch. On some occasions, the laptop was plugged directly into the downlink port of the CS2400, thereby causing the LANShield

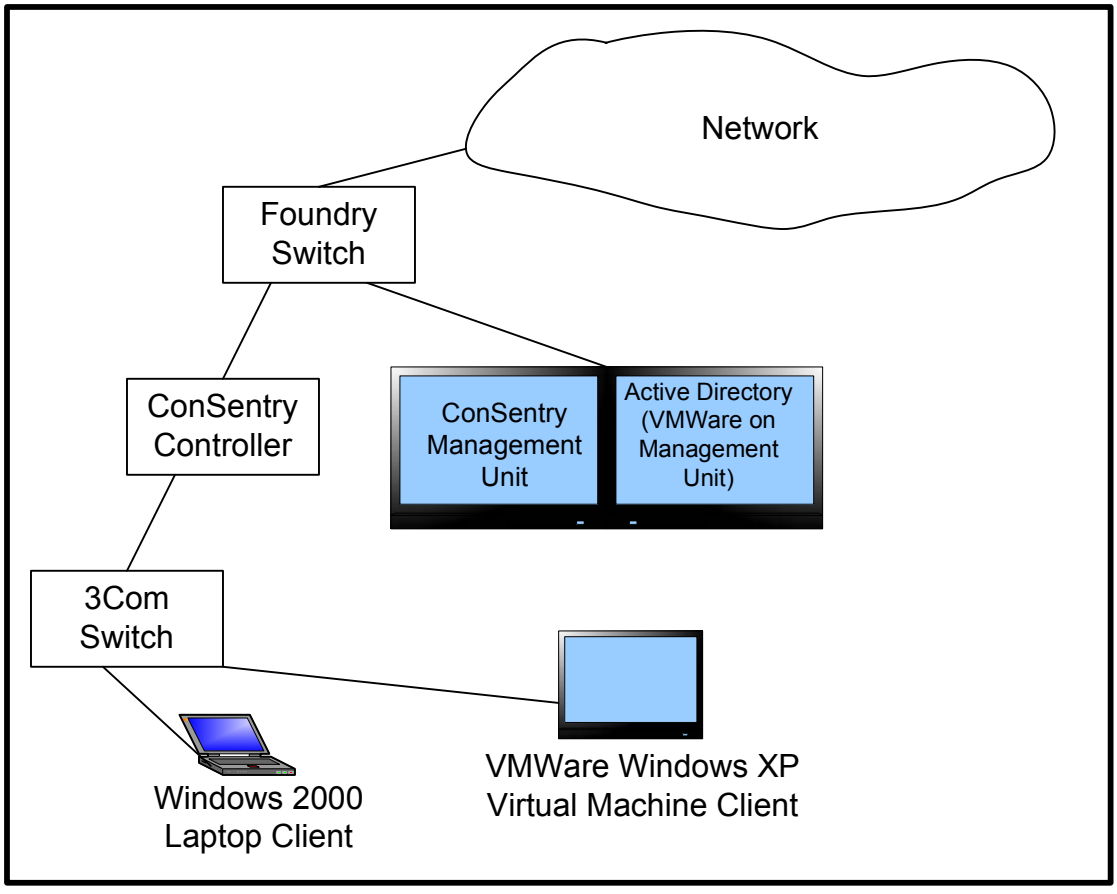

Figure 3. Testbed for ConSentry equipment.

controller to act like an edge switch. Both configurations functioned identically for our tests.

The ConSentry controller enforces policy-based authorization for users' access to services for which they have authorization. The controller enforces authorized user access by the use of Roles and Policies. The system associates each user with a Role, and each Role has one or more Policies. A Policy is a definition statement that either allows or disallows a particular activity. For instance, it may allow connections to a certain IP address or allow access to a specified application protocol. Roles specify broad categories of use, and they might include policy groups' activities such network administration, security teams, and typical users. Roles can and do provide authenticated users with varying levels of network, program, and data access.

We created a variety of users with different roles and different policies as to what actions users with that role could perform. These users were in a test domain that we established on an LDAP server. For these tests the LDAP server was running on a VMWare virtual machine executing on the ConSentry management unit. We also created a non-domain user, representative of Linux, Macintosh, or local Windows users. So we could test out cases where two clients were logged in and accessing the network at approximately the same time, we created a VMWare client system on the ConSentry management box and connected that virtual system into the 3Com (access layer) switch.

When the ConSentry equipment functioned properly, it worked well, assigning the role correctly to the various domain users and restricting or permitting network actions as specified. When the local, non-domain user logged in and tried to access the network, the ConSentry controller blocked his access, which is the desired action, since he is unauthenticated.

There is a captive portal feature on the ConSentry equipment to capture the non-domain users when they try to access the network through a browser and force them to authenticate before granting them network access. If a non-domain user attempts to perform a network access (SSH or telnet for example), he will be denied by the ConSentry because he is unauthenticated. That user must initiate a web browser, which upon any access will redirect the user to the captive portal on the ConSentry management box. The captive portal will prompt the user for his authentication credentials (e.g. username and password), and attempt to authenticate him. If the authentication is successful, the proper role is assigned to the user and appropriate access is permitted. The user can then return to his command window or screen and perform his SSH, telnet, or whatever network function he was trying to perform, providing of course, that the policies for his role permit the attempted function. If the authentication failed over several attempts, the user remains blocked from accessing the network.

While we saw the captive portal feature work once under a specific set of circumstances when the ConSentry systems engineer was installing and configuring the equipment, we have not been able to make the captive portal operate in our environment. (The browsers seem to need a proxy configuration file, which they are blocked by the ConSentry from

accessing. It appears that without the proper proxy configuration, the browser is never executing the captive portal code.)

At times, the ConSentry equipment would go into a state where it would block all access by everyone, even valid accounts. It appeared to block all traffic, including traffic to the LDAP server, so the Windows domain users could still log in using cashed Windows domain credentials, but have no network access. This condition would sometimes persist through a reload/reboot of the ConSentry controller/switch. This event occurred numerous times, but was not necessarily repeatable.

At first, it seemed that if there were no validated users on the system and a non-domain user logged in locally to the laptop connected to the network, then all subsequent network access from that laptop was blocked and remained blocked, even when a Windows domain user logged in from said laptop. (The ConSentry equipment did not seem to validate the domain user against the LDAP credentials, hence the user would be authenticated based upon previously cashed credentials.) We cleared this condition by reloading/rebooting the ConSentry controller/switch. Eventually we found that we could not consistently repeat this end result by repeating the same scenario. Also, we could not consistently clear this problem by rebooting/reloading the ConSentry controller/switch.

The ConSentry equipment has the ability to perform host remediation, or "host posture checking" in their terminology. We did not exercise this feature of the ConSentry equipment because it relies on a third party software client ("dissolvable agent"), supplied by a company on the DOE sensitive countries list. ConSentry has plans to replace this third party agent with one supplied from a company in an accepted country, although this did not occur in time for our tests.

The currently supplied ConSentry dissolvable agent is an Active X or Java applet that performs a compliance check on the host. It checks for compliant Windows Service Packs and Hotfix versions, and spyware detection, adware detection, and anti-virus software [14]. As stated above, we did not verify these claims.

The ConSentry controller predicates its protection model on authenticating users and then allowing the user to access resources for which they have authorization. This is, in essence, a client to server model where there is a person at the client device that can perform a network login. This model does not work in all cases. Many network-attached devices function autonomously. For example, a network attached printer functions on the behalf of many users, yet there is no authorized user at the printer to activate the network login. Other network devices, such as IP telephones, may not be able to support a network login, yet they still require network access. In situations where the ConSentry protection model is not appropriate, Sandia would need to devise network connection policies to support equipment such as printers and IP telephones. In testing the ConSentry controller, we found no way to allow a printer network access without disabling the network protection mechanisms. Any deployment of network protection equipment such as the ConSentry will need to address the policies and practices for controlling atypical or autonomous network devices.

We were not able to test how the ConSentry equipment handles non-interactive logins or multiple user logins from the same system. This can occur in several scenarios. Multiple users can log in to a Linux desktop, each performing network transactions. Although Fast User Switching is not available on Windows XP Professional-based computers that are part of a domain network, a local XP user may have authenticated through the ConSentry via its captive portal feature and now a different local user might log in through the Microsoft Fast User Switching feature. A user with a large or long running, but lower priority program may distribute it across a "virtual cluster" of systems, absorbing the unused compute cycles available in off hour (out-of-normal-workday) times that the machines would otherwise sit idle. Depending on where the ConSentry devices are placed in the network, and the scope or "breadth" of the virtual cluster, the ConSentry equipment could impact the operation of the virtual cluster. In these cases it is not clear that the ConSentry devices would work at all, and if so whether they would perform properly. Several questions come to mind:

- Would each user be forced into the role (be it more or less restrictive) of the first user to log in?
- Would the role be assigned based on the latest user to log in, thereby changing the set of permitted actions for the previously logged in users?
- How would ssh processes spawned for distributed or virtual cluster computing, authenticate?
- If we were using Voice over Internet Protocol (VoIP) phones, how would they authenticate and get their role assigned?

In summary, the ConSentry equipment we tested seemed to be able to perform role assignment and restricting or permitting network actions based upon that role in the case of one interactive user on a machine at a time. However, multiple concurrent users on a machine; non-interactive users, processes, and devices; and distributed or virtual cluster processing are all likely to be problematic. The stability and reliability of the equipment is questionable. At this time, it does not seem stable and consistent enough for enterprise use, especially in a network with mixed Windows domain users, local Windows users, Macintosh users, and UNIX/Linux users.

4 Network Authentication Pilot Using 802.1x

Under the Network Authentication Investigation project, Sandia implemented a small pilot using IEEE 802.1x to control network port access.

4.1 802.1x Operating Details

802.1x is an IEEE standard, and when in use, forces users to be authenticated to a network at the switch port before full network access is allowed. Within the 802.1x authentication framework there are three main parts: client, network access server (NAS, which is a switch), and a Radius server. In 802.1x terminology the client is called the supplicant, the switch is called the authenticator, and the RADIUS server is called the authenticating server. To understand 802.1x, one must understand each of these components and how they can be configured individually, as well as how they communicate to perform an authentication.

The supplicant is an application running on the end user's operating system that, in basic terms, takes the user's credentials, and requests access to the network. It then sends the user's credentials to the NAS when asked for the credentials.

The NAS controls the physical port and forwards the authenticating information to the Radius server. If the Radius server accepts or authorizes the user then the switch turns "on" the switch port and the user has access to the remainder of the network.

The Radius server performs the validation of the user's credentials to see if the user is permitted to access the network.

802.1x controls access to the network by electronically shutting down all of the switch ports to which the end users are connected. Then it turns "on" each port individually, based on a per user authentication. The port can be in one of two states at all times when 802.1x is enabled for that port: on or off. To be "on" means that the Radius server has validated the user and has granted access to the network. If the port is "off" the user cannot send any traffic over to the network except for the authentication protocol. That protocol is called Extensible Authentication Protocol (EAP), and is used by the 802.1x framework to pass the authentication information to each of the main components. A successful authentication occurs by the supplicant sending a message to the switch to start an authentication session. Then a series of access requests are followed by passing the client credentials to the Radius server for validation. Figure 4 shows how the exchange of packets occurs for a successful authentication.

EAP allows for several different forms of authentication. These include sending user name and password in clear text, in special hashes, and by using certificates. The availability of these authentication types depends on the three components of the system and their capabilities.

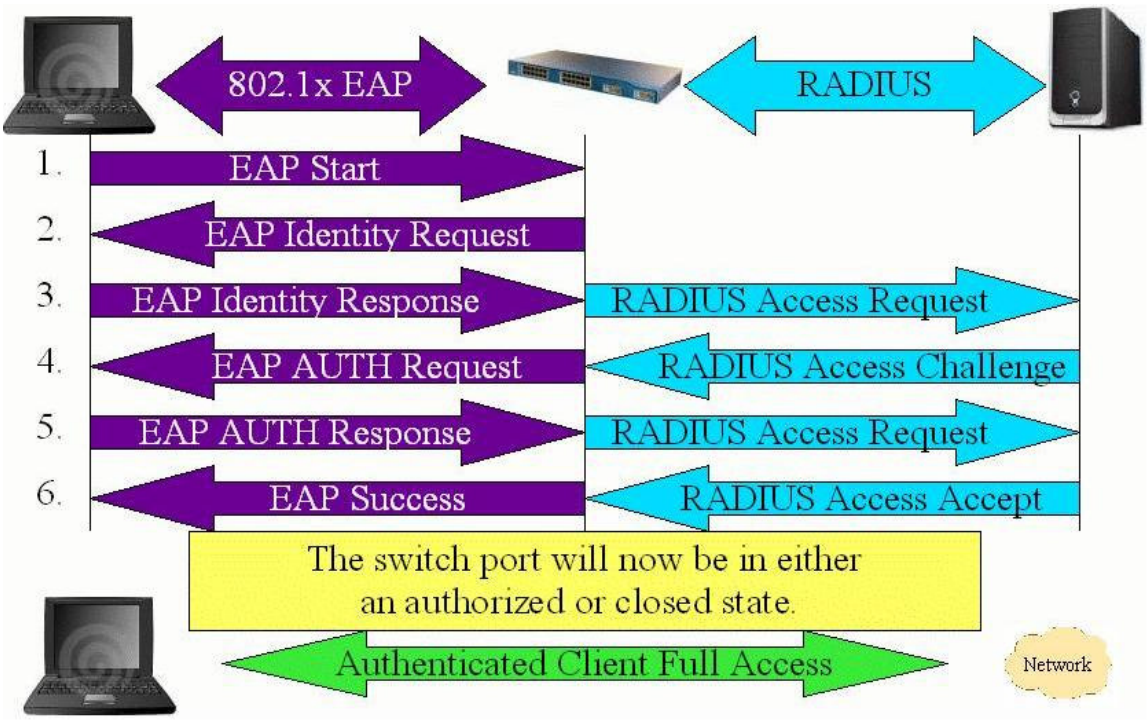

Figure 4. 802.1x authentication session.

4.2 Sandia Requirements and Goals

This pilot needed to permit authentication with the following two forms of credentials.

- Smart Card: an ID card that forces the users to type a password into a software application to gain access to the card. The card contains Public Key Infrastructure (PKI) certificates issued by a trusted Certificate Authority. In the 802.1x system this becomes the client's credentials that allow the user access to the network.
- CryptoCard: a token card that forces users to enter a personal identification number (PIN) into it to get a randomly generated password associated with the user's name. The password is also generated on a Crypto Card server that has a list of possible passwords for the user to use to authenticate to the system.

This requirement would become the driver for the entire pilot. It is important here to define a term that will be used throughout the rest of this chapter.

- **Fully Capable System**: is a system with the ability to perform both Smart Card and CryptoCard authentication for each operating system (Windows, Macintosh, and Linux) considered.

When considering how to accomplish smart card and CryptoCard authentication, the first objective was to research and decide which EAP type to use. The need for smart card and CryptoCard drove this decision because EAP defines what type of credentials will be valid for the user. The choices were:

- For Smart Card: EAP-TLS;
- For Crypto Card: EAP-PEAP-GTC and EAP-TTLS.

The following is a list of the goals created for the pilot.

- Provide at least one form of authentication for each of the operating systems;
- Find problems and inadequacies in the 802.1x system;
- Set up a test lab;
- Set up an operational pilot.

4.3 Supplicant

The supplicant, as stated before, is an application running on the user's operating system, which passes the clients credentials to the switch for authentication. Configuration of the supplicant is the simplest part of the overall 802.1x development process. Although it is the simplest part of the process it is also the most limiting factor of a fully capable 802.1x system.

4.3.1 Requirements

The first requirement is that to perform smart card authentication you must have an additional piece of software and hardware. This piece of hardware is called a smart card reader and is the physical device into which the smart card is inserted. The software needed is the drivers for the reader, which are necessary to extract the credentials from the smart card. This is one of the biggest problems in setting up a fully capable 802.1x system, and will be discussed further in a later section. In addition to needing the smart card reader, the supplicant must also provide support for TLS authentication.

To perform crypto card authentication the supplicant had to provide support for either EAP-PEAP-GTC or EAP-TTLS.

4.3.2 Client Setup

Several supplicants were set up on each of the operating systems that we wanted to support. In general most of the supplicants behave very similar. To configure a supplicant you use drop down menus (or something similar) to decide the type of EAP that you wish to use. Then you enter your user name and password at prompts to perform the authentication. There are a few nuances that are specific to each supplicant will be described in the following sections.

4.3.2.1 Windows

4.3.2.1.1 Windows Native

The Windows native supplicant is enabled by default on Windows XP. To configure the supplicant, go into the properties box (shown in Figure 5) for the Ethernet card and then

click the authentication tab. At the authentication tab, define the specific EAP type desired and then click OK. A downside to the windows native supplicant is that it only supports three EAP types, none of which are capable of supporting the CryptoCard.

Figure 5. Windows client configuration.

4.3.2.1.2 Odyssey Access Client

This supplicant has numerous capabilities, and is a much better supplicant than Windows Native. It offers support for several EAP types including those needed for smart card and CryptoCard authentication. The configuration is a little different than Windows, but generally the same. The application has to be run like any other third party application, but it is simple to configure, as illustrated in Figure 6.

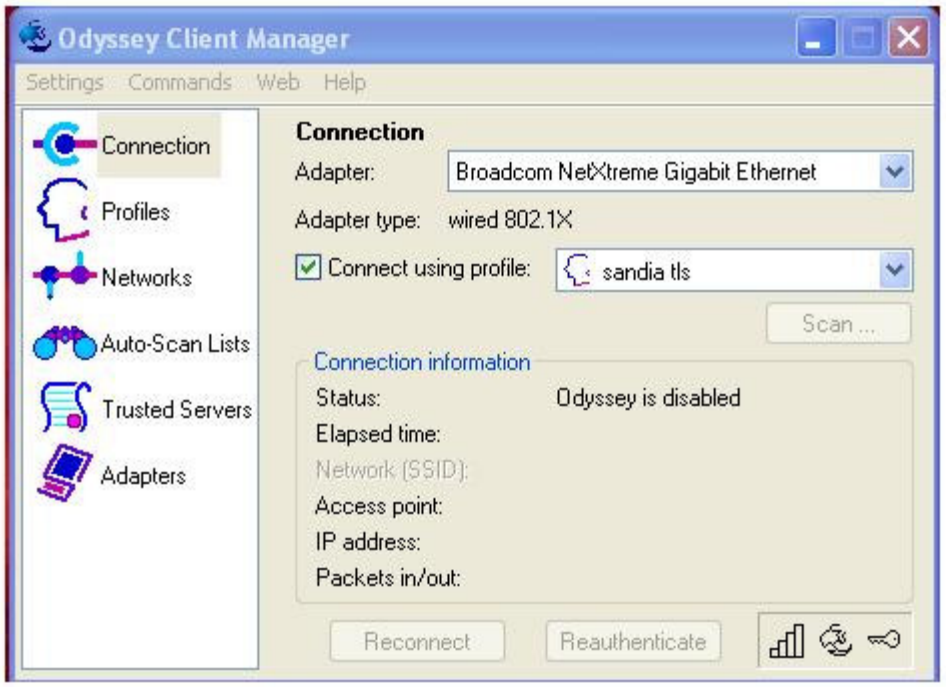

Figure 6. Odyssey client configuration.

4.3.2.2 Macintosh

4.3.2.2.1 MAC Native

This is a very capable supplicant and it behaves the same as the others do in terms of configuration. It is an application that you must run to configure like the Odyssey Client, but matches Odyssey in capabilities. It offers support for several types of EAP including support for smart cards and CryptoCard.

4.3.2.3 Linux

4.3.2.3.1 Meetinghouse Client

The Meetinghouses client behaves and is configured in much the same way as the previously mentioned supplicants. It has support for a wide range of EAP types. A small drawback is that it only works on the new Red Hat versions. A notable difference in this supplicant is that one must statically enter a password for the CyptoCard authentication. This is different from the Windows and Macintosh supplicants that prompt for user information. Having to generate a password with the CryptoCard, enter it into the static configuration, and then authenticate, may be an awkward procedure that limits user acceptance.

4.3.2.3.2 Xsupplicant

This supplicant is the only one that has a different form of configuration. To configure Xsupplicant you must modify configuration files with the correct parameters to perform the authentication. A GUI is available that is supposed to work with several different Linux distributions, but we were not able to make it function properly during this project.

4.3.3 Client Problems

When looking at a fully capable 802.1x system, the client is the most limiting factor. This is not just based on the client support for different EAP types, but includes the availability of smart card readers and drivers for the various systems.

Linux and Macintosh do not have support for the smart cards in use at Sandia. This is a problem because without the reader, the supplicant cannot get the users credentials. Without the users credentials no successful authentication can take place. The supplicants on Linux and Mac do support smart card authentication, but the lack of smart card reader support denies the ability to perform the authentication. This limitation forces the use of CryptoCards with these systems.

The Windows native supplicant does not support any EAP type that allows CryptoCard authentication. The alternative, Odyssey Client, does support the correct EAP types, but requires technical support for the third party application.

Xsupplicant is not a viable supplicant because of slow performance. During testing, the fastest authentication took four minutes. Additionally, it only performed two successful authentications among several attempts. If Xsupplicant was consistent in authentication it might be a possible solution, but the inability to work correctly coupled with the long wait time for authentication makes this option not viable.

The following chart displays the supplicant as well as the operating system's ability to perform smart card and CryptoCard authentication. As can be seen in Table 2, this is a limiting factor to implementing the fully capable system described earlier.

Table 2. Supplicant support for two factor authentication.

Client	Smart Card Support	CryptoCard Support
Windows Native	**Yes**	**No**
Odyssey Client	**Yes**	**Yes**
Mac OS	**No**	**Yes**
Xsupplicant	**No**	**Yes**
Meetinghouse Client	**No**	**Yes**

4.4 Switch

4.4.1 Technology Review And Decisions

The requirements for the switches were fairly simple in regard to their technical specifications. They had to support 802.1x authentication. An additional specification was that the switches in the testbed needed to be representative of what Sandia has in its existing environment. A Cisco 3750 and a Foundry FES 2402 Prem were purchased to do the work. Several non-802.1x-capable commodity switches were obtained for multiple host environment (see Section 4.4.3) and other advanced testing.

4.4.2 Configuration

The configuration of both switches was straightforward. It took roughly ten commands to get one port enabled for 802.1x, with about thirty extra seconds per port on one switch. A list of the commands, for each switch, can be found in Tables 3 and 4.

Table 3. 802.1x configuration commands for Cisco switch.

```
Switch# configure terminal
Switch(config)# aaa new-model
Switch(config)# aaa authentication dot1x default group radius
Switch(config)# dot1x system-auth-control
Switch(config)# radius-server host 192.168.1.1 auth-port 1645 acct-port 1646 key MySecreT
Switch(config)# interface g1/0/1
Switch(config-if)# switchport mode access
Switch(config-if)# switchport access vlan 1
Switch(config-if)# dot1x port-control auto
Switch(config-if)# end
```

Table 4. 802.1x configuration commands for Foundry switch.

```
switch(config)# aaa authentication dot1x default radius
switch(config)# radius-server host 192.168.1.1 auth-port 1645 acct-port 1646 default key
            MyKeY dot1x
switch(config)# dot1x-enable
switch(config-dot1x)# enable ethernet 13 to 24
switch(config-dot1x)# exit
switch(config)# interface e 13
switch(config-if-13)# dot1x port-control auto
```

(Apply the last two commands for all interfaces that are to be 802.1x interfaces)

The Cisco switch worked correctly right away with no difficulties. The Foundry switch proved to be a little bit more difficult to set up because the Foundry code version was out of date and it needed to be upgraded. Once the newer code version was loaded on the Foundry, it worked fine.

4.4.3 Multi-Host Problem

The main concern pertaining to the switches is their ability to support a multiple host environment. A multiple host environment occurs when there is a switch port enabled with 802.1x authentication, and a user connects an office switch (without 802.1x enabled) to that port because they want to use several computers on the one Internet connection. The problem lies in how the authentication will occur through a switch that is not 802.1x enabled, and if a user will be allowed to access the network at all. The problem is illustrated in Figure 7.

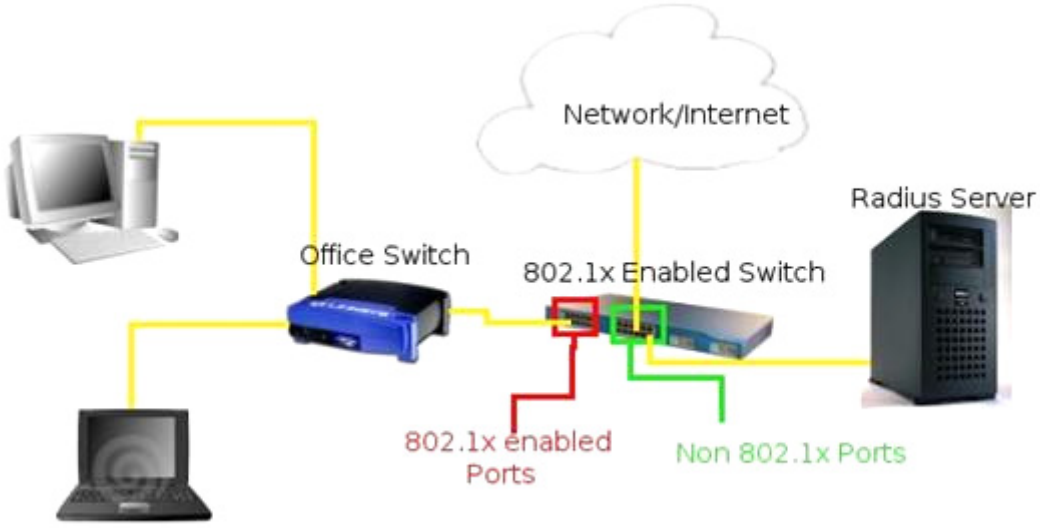

Figure 7. Multiple host environment.

Foundry's newer code version supports the ability to force authentication of each host, which is what is needed for the 802.1x system. The Cisco does not allow the same capability. The Cisco switch forces one of the hosts to authenticate and once that occurs, it allows any other users plugged into the office switch access to the network. Another problem is that if the office switch is "too smart" it will not forward the 802.1x packets correctly. So, the office switch must be a dumb switch to allow the proper functionality, even with the Foundry switch as the 802.1x enabled switch in the configuration.

4.5 RADIUS

The Radius server was the most complicated and time consuming part of establishing the pilot. As stated before it is the component of the system that validates the users' credentials. It can do this in numerous ways, depending on the specific server that is chosen. When looking at the Radius servers it was important to keep in mind the concept of a fully capable system. To have the fully capable system, the Radius server had to support both smart cards and CryptoCards. Radius servers are able to perform several different types of authentication by default. They differ in their individual

implementations, but most support the necessary types of authentication for smart card and CryptoCard authentication.

4.5.1 Technology Review And Decisions

There are several different Radius servers available. When looking at Radius servers, we found that Sandia had a site license for Radiator, a specific Radius server. After configuring Radiator, some time was also spent configuring Free Radius, another Radius server that is free software anyone can use. Free Radius was also configured because it appeared to perform some specific aspects of the CryptoCard authentication better than Radiator. Experience was gained in configuring both Radius servers, but Radiator was chosen because the configuration for CryptoCard authentication was accomplished on Radiator before Free Radius, allowing us to work on other aspects of the pilot.

4.5.2 Smart Card Authentication Setup

Smart card authentication uses EAP-TLS, as stated earlier. To get TLS working one must point attributes within the Radiator configuration file to necessary certificate files for validation of the users' credentials. Understanding how certificates are validated is out of the scope of this document.

Setting up the Radiator configuration file was trivial, but getting the authentication to work was difficult because of the nature of certificates. Certificates are very specific in how they are set up. If one piece of the configuration is not exactly where it should be, the certificates will not work. Although Radiator was set up quickly, it took much time and effort to get the certificates working properly. After the certificate system was working, half of the fully capable system (in terms of Radius configuration) was set up.

4.5.3 Crypto Card Authentication Setup

The crypto card setup was the most time consuming and toughest part of the system to configure properly. The CryptoCard had to authenticate through the Radius server to the crypto card server. This was not an easy task, because the crypto card server does not have any easy way for the authenticator to connect to it, without custom code. The EAP types chosen for the crypto card authentication were EAP-PEAP-GTC and EAP-TTLS.

Sandia has a Free Radius server in existence that already connects to the crypto card server. During configuration it was discovered that the Free Radius code was too customized to recreate on our testbed servers in a time frame acceptable for the pilot. For the pilot, the users' credentials were authenticated through Radiator to an LDAP server that performed its authentication against the CryptoCard server. The back end portion of the Radius configuration was complete and we had a fully capable system, in terms of the Radius component.

4.6 Setting up 802.1x for New Users

Setting up the 802.1x system for new users is an easy task. The authentication server is already deployed and operating. To set up individual users, the following steps are performed.

- Find the port on the edge switch into which the user is connected;
- If the switch does not have 802.1x enabled on it already, issue the commands to turn it on and then define the correct Radius server to use (this takes roughly 4 commands);
- Apply 802.1x authentication to the port into which the user is connected;
- Set the port to 802.1x auto mode;
- Configure the users' supplicant for 802.1x.

Turning on 802.1x after finding the switch and port into which the user is connected, takes between five to ten minutes.

4.7 Pilot Status

The pilot is currently deployed to four users, and has been in this state since July 24, 2006. From its creation the pilot has seen stability and success with only a few problems occurring, which will be discussed in a later section. Although the pilot has been running consistently, it is still not deployed to an ample enough user base to get a true understanding of how the 802.1x system would work on a larger scale.

Upon starting the pilot there were some questions on the ability of 802.1x to allow for a multiple host environment (section 4.4.3). This issue has been worked out for the Foundry switch and the multiple host environment has worked as it should. How Cisco handles this environment, is still an open problem.

Other positive results include the ease of use of the clients and the ability to stay logged into the 802.1x system while having a locked workstation.

4.8 Problems & Solutions

The following specific problems have occurred:

- Problem: When attempting to authenticate with a smart card, a user will not be able to authenticate.
    - Solution: The log files on the Radius server showed that the problem with the authentication attempt was that the client/supplicant was not getting the certificate off of the smart card. The problem has no apparent fix besides logging off of windows and logging back on. It appears as though there is a bug in either the client/supplicant or the smart card reading software, which impedes the opportunity to fix the problem. With it only happening once it seems as though there may have been unseen circumstances that might have affected the authentication attempt, and

unless the issue recurs on a larger scale, it should pose no threat to the capability of the 802.1x system.

- Problem: The back end for a crypto card authentication failed once, and rejected the authentication attempt.
    - Solution: The problem lies somewhere with the user's credentials and the LDAP server. It was not a fault of the 802.1x system but rather of that specific user and the crypto card authentication server. Attempts were made by different users at around the same time and their credentials were validated.

- Problem: Sandia Macintosh computers are set up to first authenticate against the Kerberos server before logging into the computer. For 802.1x to work properly the user must first login locally to their machine and then they are able to authenticate to the 802.1x system. The problem is that when a network is present the Kerberos settings on the Mac do not allow for the user to log in locally first, hence, denying them the ability to login into the network at all.
    - This is a problem with the default login settings for the Sandia Macs because they require Kerberos authentication first, and will only allow a local login if there is no network cable plugged into the computer. A solution has yet to be found for allowing a local login before the network login. A possible solution is changing the settings in the /etc/authorization file to allow for this, but that has not yet been accomplished.

There remain problems inherent to 802.1x, such as uniform client support for the commonly used operating systems at Sandia, and the manner in which Cisco handles a multiple host environment.

Another problem is overall support for 802.1x authentication in devices such as printers, palm pilots, VoIP phones, etc. These were not addressed in this pilot.

4.9 Pilot Conclusions and Path Forward

An IEEE 802.1x authentication framework has been established that allowed for one form of authentication on each of the main operating systems under consideration. Problem areas were located (and to some extent, remedied) within the current state of an 802.1x system. A pilot was initiated and used for several months.

The pilot explored, and in most cases got much better performance from third party software applications. This takes away the myth that to have a fully capable 802.1x system you must use all Cisco or all Microsoft products. By being able to use the other pieces you can have more options and are not tied to one vendor to supply each component of the system.

The pilot is currently deployed to a small user base. The 802.1x framework in its current state cannot provide a fully capable system, but it is in a state that could be implemented

to provide a greater level of security to Sandia networks. The pilot did not collect statistics to measure the effects, if any, of 802.1x on switch throughput and network performance.

A possible path forward is to extend the reach of the pilot and set up 802.1x authentication for more users. This will enable the Network Authentication team to more accurately assess the outcomes of a more extensive pilot, and discover how 802.1x will function on a larger scale. This would also provide an opportunity to continue exploring and investigating the problems with 802.1x and researching possible solutions.

5 Recommended Strategy for Network Authentication at Sandia

Sandia's business operation increasingly relies on information technology and data stored electronically. However, that technology and data are under constant threat of attack. Perpetrators are getting faster at introducing new network attacks, and the attacks themselves are more frequent. Therefore, detection and containment at the edge of the network is vital. With more emphasis on technology, constantly increasing threats, and declining budgets, Sandia needs a cost-effective way to protect their network resources. Network authentication provides a level of protection.

Any network authentication technology should increase the security of our networks and increase user accountability. It should be a component of a larger data security posture that coordinates with future planning and road mapping efforts. Preferably it should incorporate some sort of two-factor authentication, be convenient, be easy to maintain and support, be reliable, be inexpensive, and work across platforms for both office and mobile use. It must be manageable and scaleable.

There are five technology functions that are accepted and expected as part of a Network Authentication and Control (NAC) product [2], although these five functions are only vaguely defined. They are:

- Pre-connect host posture assessment;
- Host quarantine and remediation;
- Network access control based on user identity;
- Network resource control based on identity and policy;
- Ongoing threat analysis and containment.

Many NAC products do a good job of fulfilling one or more of these functions, but none of the various products covers all five functions well. ConSentry does well with regard to access control and identity-based resource control. Microsoft's forthcoming product is expected to do well with posture assessment and quarantine/remediation. To cover all functions well, will probably require the use and integration of several products.

On this project we determined a set of features we feel are necessary as part of our overall security posture. Not all features will be practical on every system or device.

- Pre-connect host posture assessment, quarantine, and remediation (This should include checking for the proper patch levels of the operating system, correct containment/countermeasure software is installed and running, and correcting deficiencies. This should also inspect systems attempting connection for actively executing known viruses.);
- Ongoing host posture assessment (to ensure containment/countermeasure software is not terminated during a connection session);
- Network access control (Either user identity based or device based or both, if practical.);
- Network resource control;

- Containment/countermeasure software (e.g. firewall, virus protection, anti-spyware, intrusion detection, etc.).

Sandia is already performing elements of posture assessment, identity-based network access control, and network resource control, but more could be done. The situation should be analyzed to determine if it is necessary, useful, and cost effective to do more in these areas. Sandia already deploys containment/countermeasure software as part of the Common Operating Environment for Microsoft Windows systems. A future project may wish to examine the possibility of adding intrusion detection software to the individual systems.

To tighten Sandia's security posture in any of the areas listed above, analysis will be needed to determine what it will take to accomplish the desired task. This analysis should include:

- How much will it cost and how will it be funded;
- A cost/benefit analysis;
- What role will this task or feature play in Sandia's overall security posture;
- Can the project be completed quickly enough to make a difference;
- If the project spans fiscal years, will continued funding be available;
- Reliability of the solution;
- Any vulnerabilities introduced by the solution;
- Day-to-day operations;
- Any other organizational impacts introduced by the solution.

This project determined that IEEE 802.1x does work and is possible to use with Sandia's network infrastructure. The project was able to successfully test 802.1x with two-factor authentication. It used both CryptoCards and Smart Cards. However, each card is only useful on certain host operating systems and in certain use cases. Further research is required to reduce the complexity of two-factor authentication. The complexity is also an issue that affects system wide reliability. Any project that addresses wide spread availability will need to consider authentication reliability. With 802.1x, or any network access control system, any single failure in the authentication mechanism effectively disables the entire network.

802.1x may be useful to Sandia in the future for switch port control, but is most immediately useful as part of the IEEE 802.11i standard for wireless network security. Limited deployment of 802.1x, via 802.1x enabled switches, may also be of immediate use to secure exposed wired ports in locations that are difficult to physically control or secure.

Sandia should pursue investigation of more tools to effect host posture checking and client remediation. This should include both the pre-connect host posture assessment, quarantine, and remediation, and the ongoing host posture assessment as defined above.

Appendix A Bibliography

1. D. W. Chadwick, *Understanding X.500 -- The Directory*, http://sec.cs.kent.ac.uk/x500book, 1996.

2. Joel Conover, NAC Vendors Square Off, in *Network Computing*, July 6, 2006. Available online: http://www.networkcomputing.com/channels/networkinfrastructure/showArticle.jhtml?articleID=189602326.

3. Sean Convery, *Network Security Architectures*, Cisco Press, Indianapolis, IN, 2004.

4. Dorothy Elizabeth Robling Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, MA, 1982.

5. Jon Edney and William A. Arbaugh, *Real 802.11 Security*, Addison-Wesley, Boston, 2004.

6. Eric A. Fisch and Gregory B. White, *Secure Computers and Networks*, CRC Press, Boca Raton, FL, 2000.

7. Æleen Frisch, *Essential System Administration*, 2nd ed., O'Reilly, Sebastopol, CA, 1995

8. David K. Hsiao, Douglas S. Kerr, and Stuart E. Madnick, *Computer Security*, Academic Press, New York, 1979.

9. "Implementing Network Admission Control Phase One Configuration and Deployment", Cisco Systems, Inc., 2005.

10. "Kerberos: The Network Authentication Protocol," http://web.mit.edu/Kerberos/www, December 6, 2005.

11. R. Langston, "Network Access Control Technologies", white paper, The Business Forum. Available online: http://www.bizforum.org/whitepapers/sygate-4.htm.

12. Naval Research Laboratory, http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html, August 18, 2000.

13. R. M. Needham and M. D. Schroeder, Using Encryption for Authentication in Large Networks of Computers, in *Communications of the ACM*, vol. 21, pp. 993-99, December 1978.

14. “Network Admission Control”, ConSentry Networks Technology Solution Brief, Available online: http://www.consentry.com/download/ConSentryNAC7-21.pdf, 2006.

15. Open System Consultants, http://www.open.com.au/radiator/technical.html#wireless.

16. Port-Based Network Access Control, IEEE STD 802.1X-2001, Institute of Electrical and Electronic Engineers, Piscataway, NJ, 2001.

17. Greg Shipley, Network Node Validators: Catching Rogue Nodes, in *Network Computing*, vol. 16, November 10, 2005.

18. “Symantec Posture Plug-in for Cisco Network Admission Control (NAC) 2.0”, Symantec.

19. “TCG Trusted Network Connect TNC Architecture for Interoperability”, Trusted Computing Group, May 3, 2005.

20. Wikipedia, *Extensible Authentication Protocol*, http://en.wikipedia.org/wiki/Extensible_authentication_protocol, January 22, 2006.

21. Wikipedia, *Lightweight Directory Access Protocol*, http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol, January 28, 2006.

DISTRIBUTION:

1	MS 0662	T. Klitsner, 4341
1	MS 0662	J.K. Perich, 4343
1	MS 0672	L.G. Pierson, 5616
1	MS 0788	J.L. Akins, 4336
1	MS 0788	J.H. Maestas, 4336
1	MS 0788	P.L. Manke, 4338
1	MS 0795	P.C.R. Jones, 4317
1	MS 0799	M.J. Benson, 4318
1	MS 0799	D. Eichert, 4318
1	MS 0799	G.E. Connor, 4334
1	MS 0801	R.W. Leland, 4300
1	MS 0801	D.S. Rarick, 4310
1	MS 0806	L.F. Tolendino, 4334
1	MS 0806	J.P Brenkosh, 4336
3	MS 0806	N. Dautenhahn, 4336
3	MS 0806	J.M. Eldridge, 4336
1	MS 0806	A. Ganti, 4336
1	MS 0806	S.A. Gossage, 4336
1	MS 0806	C.M. Keliiaa, 4336
1	MS 0806	B.R. Kellogg, 4336
1	MS 0806	L.G. Martinez, 4336
1	MS 0806	M.M. Miller, 4336
1	MS 0806	J.H. Naegle, 4336
1	MS 0806	U. Onunkwo, 4336
1	MS 0806	J.A. Schutt, 4336
1	MS 0806	L. Stans, 4336
1	MS 0806	J.S. Wertz, 4336
1	MS 0806	D.J. Wiener, 4336
5	MS 0806	E.L. Witzke, 4336
1	MS 0806	T.J. Pratt, 4338
1	MS 0806	T.C. Hu, 4338
1	MS 0806	T.D. Tarman, 5622
1	MS 0807	J.F. Mareda, 4537
1	MS 0813	R.M. Cahoon, 4311
1	MS 0813	J.P. Abbott, 4312
1	MS 0813	C.D. Brown, 4312
1	MS 0813	J.P. Long, 4312
1	MS 0813	G.K. Rogers, 4312
1	MS 0813	D.R. Wachdorf, 4312
1	MS 0832	J.H. Dexter, 4335
1	MS 1393	J.A. Larson, 12120
1	MS 9012	R.D. Gay, 8949
1	MS 9012	B.A. Maxwell, 8949
1	MS 9012	M.G. Mitchell, 8949

2	MS 9018	Central Technical Files, 8944
2	MS 0899	Technical Library, 4536