San Francisco State University
Independent Study in Advanced Networks
[BICS 699]
Dr. Sameer Verma [sverma@sfsu.edu]

Philie Chang
philiec@yahoo.com

Herbert Ishii
hishii@yahoo.com

Implementing Secure Services over a Wireless Network

A Captive Portal and Traffic Shaping Approach

About this document

This report is the result of an independent study course (BICS 699) conducted in the Information Systems and Business Analysis department in the College of Business at San Francisco State University. The course was taken by P. Chang and H. Ishii and was under the guidance of Dr. Sameer Verma.

Table of Contents

Abstract

IMPLEMENTING SECURE SERVICES OVER A WIRELESS NETWORK:

A CAPTIVE PORTAL AND TRAFFIC SHAPING APPROACH

Abstract

The concept of IEEE 802.11b standard-based wireless network or "Wi-Fi" has become very popular in the last year or so. Wi-Fi's widespread use raises questions about the notion of security of a technology that uses radio waves for communication. The primary focus of this project is the issue of authentication, authorization and possibly, accounting on a Wi-Fi network. The project does not particularly address the notion of security via encryption of data (such as WEP) transmitted wirelessly. After examining the possible options, we discuss a solution and the implementation of an experimental wireless network service at San Francisco State University.

Wireless networks are not secure at the physical layer, because unauthorized users do not need access to an Ethernet jack. They can tap into a wireless LAN using limited information and packet sniffing tools in the proximity of a network. The only form of encryption natively available on such networks is called Wired Equivalency Privacy, or WEP. It has been demonstrated that encryption keys can be sniffed and decrypted after monitoring 100MB to 1GB of traffic. Therefore, some sort of authentication is needed to ensure that acceptable use policies (AUP) are not being violated. To this end, we analyze the problem at hand and use an existing collection of tools to build a proof-of-concepts system. Our approach to a secure service is based on an open source solution called NoCatAuth. This solution allows us to implement a Captive Portal with traffic-shaping capabilities on our wireless network. Although NoCatAuth is an open source solution that is operational, it is in the beta phase. Availability of source code allows us to better understand the underlying system.

Introduction:

A new networking standard protocol labeled IEEE 802.11b is proving to be one of the fastest and most convenient ways of getting on a network. Often called Wi-Fi, it does away with the need for cabling, network jack installation, and the addition of ports on switches & hubs to add clients to a network.

Wi-Fi operates in the consumer oriented unlicensed 2.4 Gigahertz wireless spectrum, and has a useful range of about 300 to 500 feet (Panko, 2001). Maximum theoretical throughput is 11 Megabits per second.

Our goal is to implement secure services over a wireless network. Our requirements are:

- Authentication – Only provide service to clients specified in a database
- Easy Manageability – Minimize complicated management
- Traffic Shaping – Provide some Quality of Service
- Easy Access – Eliminate the need for an additional client software
- User Friendly – Easy learning curve for users
- Low Cost – Use in-house resources. This is an interim solution until a newer project (perhaps based on IEEE 802.1X) is implemented

To serve most of the requirements above, we find NoCatAuth (Flickenger, 2002), an open source captive portal solution to address many of the concerns. Some modifications and additions were made by us to tailor the system to our needs[1].

[1]Since NoCatAuth is available in the opensource realm, and is released to the community under the GNU General Public License (GPL), we made our contribution available to the community in accordance with the spirit and letter of GPL.

Background

A. Notion of Security

A brief overview of the 7-layer OSI Reference model of network communication is in order (Panko, 2001).

1. Physical Layer – Defines the mechanical and electrical specifications of the network medium and network interface hardware, how they connect with one another, and how data is placed on and retrieved from the network medium.

2. Data Link Layer – Organizes the Physical layer's 1s and 0s into frames. The Data Link Layer also detects and corrects errors; controls data flow, and identifies particular computers on the network.

3. Network Layer – Moves information across a network made up of multiple network segments. The Network layer does this by examining the destination Network address and sending the packet to the next transit point in the internetwork.

4. Transport Layer – Ensures reliable data delivery through a variety of mechanisms, like orderly connection establishment and teardown, acknowledgement messages, sequence numbers, and flow control.

5. Session Layer – Adds control mechanisms to the data the establish, maintain, synchronize, and maintain dialog between communicating applications.

6. Presentation Layer – Transforms data into a mutually agreed-upon format that can be understood by each application and by the computers they run on. The presentation layer may also compress, expand, encrypt and decrypt data.

7. Application Layer – Specifies the communication interface with the user and manages communication between computer applications.

Wireless Networks (WLAN's) are not secure at the physical layer, because unauthorized hackers only need a laptop, a directional antenna, and a wireless card that is compliant with the IEEE 802.11b standard. Tapping into a WLAN only requires an individual to be in the general proximity of an Access Point. For instance, one can walk around any downtown district with a laptop to pick up a signal(s) from Access Points installed in office buildings (Gomes, 2001). Peter Shipley has presented results of 'WarDriving' in the San Francisco Bay Area, where he drove around town with a GPS, laptop, antennas, and an 802.11b network card and make note of all the 802.11b networks he 'stumbled' upon (Shipley, 2001). Wired Equivalent Privacy (WEP) is the only standard to provide physical & data link layer privacy. In the next section we will discuss the flaws in WEP.

B. Purpose of Encryption, WEP and other Approaches

Data is encrypted to ensure that only authorized users have access to information. 802.11b presents an interesting problem in terms of security. Because radio waves cannot be controlled effectively or enclosed by walls, additional methods of security are needed to prevent unauthorized users.

Two methods of data encryption are available for the 802.11b protocol. The first method provides 40 bit keys, called WEP or Wired Equivalent Privacy (Fennelly, 2001). Unfortunately, WEP has a serious flaw in its implementation. The keys are static – and can be sniffed in a relatively short amount of time (Verton, 2001). According to www.extremetech.com, AirSnort, a program that runs on a Linux system with a 2.4 kernel and Prism-based wireless cards, can discover a WEP key after passively monitoring a wireless network (Fisher, 2001). According to the site (http://airsnort.sourceforge.net), AirSnort can determine the WEP key in seconds after "listening" to 100MB-1GB of traffic. The second method is similar to 40 bit WEP but is operates under 128 bit keys. Unfortunately, 128-bit encryption is proprietary through vendors like Cisco, 3Com, and Lucent, and has similar weaknesses like WEP.

Some other approaches to security are:

- Radius Authentication – Very effective, but requires a RADIUS system implemented in the network
- Adding a DMZ (De-Militarized Zone) – Also effective, but adds an extra layer to manage
- Forcing VPN – Forces wireless users to create a VPN tunnel to the network

C. Need for Authentication: using SSL

According to the Internet-Draft on Secure Socket Layer (also called TLS or Transport Layer Security), the primary goal of the SSL Protocol is to provide privacy and reliability between two communicating applications. The protocol is composed of two layers (Dierks & Allen, 1999). At the lowest level, layered on top of some reliable transport protocol (e.g., TCP), is the SSL Record Protocol. The SSL Record Protocol is used for encapsulation of various higher-level protocols. One such encapsulated protocol, the SSL Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. One advantage of SSL is that it is application protocol independent. A higher-level protocol can layer on top of the SSL Protocol transparently. The SSL protocol provides connection security that has three basic properties:

- The connection is private. Encryption is used after an initial handshake to define a secret key. Symmetric cryptography is used for data encryption (e.g., DES, RC4, etc.)
- The peer's identity can be authenticated using asymmetric, or public key cryptography (e.g., RSA, DSS, etc.)
- The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions (e.g., SHA, MD5, etc.) are used for MAC computations.

D. Traffic Shaping and Quality of Service

Traffic Shaping is the general term given to a broad range of techniques designed to enforce prioritization policies on the transmission of data over a network link. Traffic shaping allows the implementation of a specific policy that alters the way in which data is queued for transmission. The methods of Quality of Service (QoS) supported in the Linux kernel (Radhakrishnan, 2001)are:

- Class Based Queue (CBQ)
- Token Bucket Flow (TBF)
- Clark-Shenker-Zhang (CSZ)
- First In First Out (FIFO)
- Priority
- Traffic Equalizer (TEQL)
- Stochastic Fair Queuing (SFQ)
- Asynchronous Transfer Mode (ATM)
- Random Early Detection (RED)
- Generalized RED (GRED)
- Diff-Serv Marker (DS_MARK)

Generally speaking, QoS is used to describe an end-to-end guarantee of some kind of quality of service. It is the idea that transmission rates, error rates, and other characteristics can be measured,

improved, and, to some extent, guaranteed in advance. QoS is needed when receiving audio (VoIP) and video (Steaming Media) transmissions, where data is flowing continuously. Quality of Service can only be achieved where some sort of Service Level Agreement (SLA) is made, otherwise it would only be “best effort” (Taylor & Hettick, 2001).

This guarantee is generally created using reservation protocols such as RSVP (Armitage, 2000) (Braden, Zhang, Berson, Herzog, & Jamin, 1997). The reservation of bandwidth and throughput is done between the client and server with several hops in between. All hopping stations (routers) have an agreement to provide the reserved bandwidth and throughput.

E. Netfilter: Working with the IP Layer

Netfilter is a set of rules that are set in the IP layer using IPTables in Linux. The relationship between Netfilter and IPTables is that Netfilter is like the Kernel in Linux and IPTables is the user-space program to interface with the kernel and provide what the kernel needs (Russell, 2001). It has four parts.

1. Firstly, each protocol defines "hooks" which are well-defined points in a packet's traversal of that protocol stack. At each of these points, the protocol will call the netfilter framework with the packet and the hook number.

2. Secondly, parts of the kernel can register to listen to the different hooks for each protocol. So when a packet is passed to the netfilter framework, it checks to see if anyone has registered for that protocol and hook; if so, they each get a chance to examine (and possibly alter) the packet in order, then discard the packet, allow it to pass, tell netfilter to forget about the packet, or ask netfilter to queue the packet for userspace.

3. The third part is the packets that have been queued for sending to userspace; these packets are handled asynchronously.

4. The final part consists of comments in the code and documentation.

   Main Features

   - Stateful packet filtering (connection tracking)
   - Many kinds of network address translation
   - Flexible and extensible infrastructure
   - Large number of additional features as patches

   Netfilter/IPTables use

   - Build Internet firewalls based on stateless and stateful packet filtering
   - Use NAT and masquerading where we don't have enough addresses
   - Use NAT for implementing transparent proxies
   - Aid the tc+iproute2 system used to build sophisticated QoS routers
   - Do further packet manipulation (mangling) like altering the TOS field of the IP header

Approach

Our approach in implementing a secure wireless network used Linux as a platform, where different kinds of services run on it. Such services will include Gateway, Authentication, and Netfilter. NoCatAuth was chosen as the open source solution that is currently at its beta stage, but has a promising future. NoCatAuth can be downloaded from: http://nocat.net/ (Flickenger, 2002)

A. Use a wireless access point as the main point of entry.

A wireless access point was used to communicate with other wireless users on the WLAN. The AP will act as the main point of entry for every user, and then it will redirect its traffic to the Gateway Server.

B. Redirect traffic to a Gateway, implementing netfilter.

Once traffic has been redirected from the Access Point to the gateway and the user is authenticated successfully, a Netfilter profile is implemented based on the user type. Such users could be divided into three classes such as: faculty, student, or administrator.

C. Use an SSL server for authentication front-end.

SSL with the Apache web server was implemented in order to secure transmission of login and password information between the client and the server. This is to prevent unauthorized users from sniffing usernames and passwords that would otherwise be transmitted as clear text.

D. Use a database for account management.

MYSQL was used to store the user account information. Data manipulation can be done via simple forms. For advanced DB manipulation, we used PHPMyAdmin.

E. Use PGP/GnuPG for Gateway<->AuthServ authentication

Communication between the Gateway and the AuthServ requires trust. PGP/GnuPG was used to secure the communication, through the use of trusted keys. The original NoCatAuth distribution specifies that the AuthServ and the Gateway reside in two separate systems, thus the need for secure authentication. Because our Gateway implementation contains both the AuthServ and the Gateway in the same box this extra layer of authentication is somewhat redundant. However, this approach allows us to migrate the AuthServ to another location very easily.

F. Use SSL for End-User<->AuthServ authentication

Secure Sockets Layer was used to guarantee the transaction between the End-User and the Auth Server. Again, this is to prevent unauthorized users from sniffing usernames and passwords that would otherwise be transmitted over wireless signals.

G. Use Gateway with Netfilter and Traffic Shaping to implement QoS

In our project, we are referring to QoS from a very simple perspective. Instead of looking at a reservation across several hopping stations, we are merely looking at some control of bandwidth and throughput between two interfaces (eth0 and eth1) on the same computer. Details of the various approaches to QoS can be found in a book titled "Quality of service in IP networks by Grenville Armitage" (see references).

For this project we chose to implement QoS features as a possible option. This option is implemented via the Class Based Queue (CBQ) method. Let's look further into what CBQ really means for a wireless LAN. CBQ is a method that works with how a router handles arriving packets at the IP layer. CBQ allows for the creation of classes. Based on markers such as source IP address, destination IP address, protocols, Type of Service (ToS) bits etc., CBQ can create a set of classes. These classes could be constructed to differentiate TCP vs. UDP traffic. For example, TCP packets that come in for an FTP transfer would get marked differently than UDP packets for streaming audio, video or real-time multiplayer games. Another way to differentiate is to create classes for traffic coming from different parts of the network or from different gateway devices. Yet another possibility is to create classes based on some user-defined characteristics that are controlled by network administrators.

Looking further into CBQ, we find that each class can get its predetermined bandwidth and under some conditions, a class may borrow bandwidth from other classes if the network administrator allows us for such a transfer (Radhakrishnan, 2001). Of course, the borrowing of bandwidth can only occur if bandwidth is actually available in the other classes. One of the strengths of CBQ is in being able to allocate link bandwidth to classes while independently assigning priorities to those classes. Thus, a router could have a high-priority real-time and a lower-priority non-real-time class, each with a different bandwidth allocation, and the packets from the real-time class would receive priority scheduling as long as sufficient bandwidth was available, or in times of congestion, as long as the arrival rate for that class did not exceed its allocated bandwidth.

NoCatAuth implements CBQ via the “throttle.fw” script in the bin folder. Looking closely and this script we see the implementation of a CBQ system using the tc package.

The general syntax for using the tc package is:

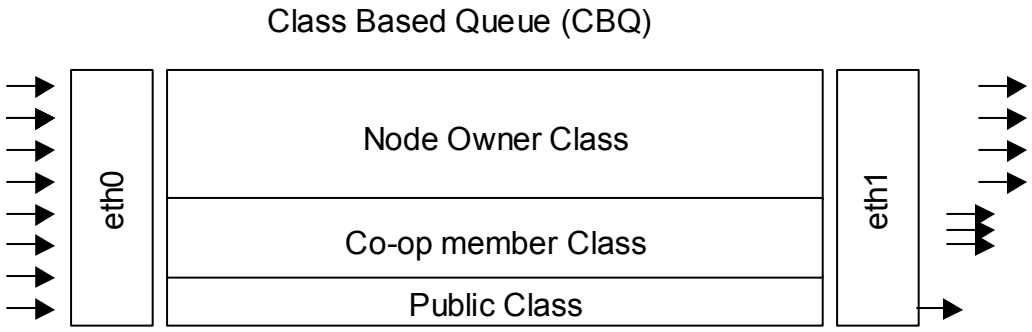
```
tc [ OPTIONS ] OBJECT { COMMAND | help }

where  OBJECT:= { qdisc | class | filter }

OPTIONS:= { -s[statistics] | -d[details] | -r[rawq] }
```

The following statement from the throttle.fw file is an example of a class based queue system being implemented on the internal device ($InternalDevice). Looking at this line we see that the throttle script is using the CBQ method for controlling the use of bandwidth in NoCatAuth.

```
tc qdisc add dev $InternalDevice root handle 10: cbq bandwidth 10Mbit avpkt 1000
```

According to the original design of NoCatAuth, the users can be put into three different classes namely owners, co-op users, and public users. Instead of using a scheme where

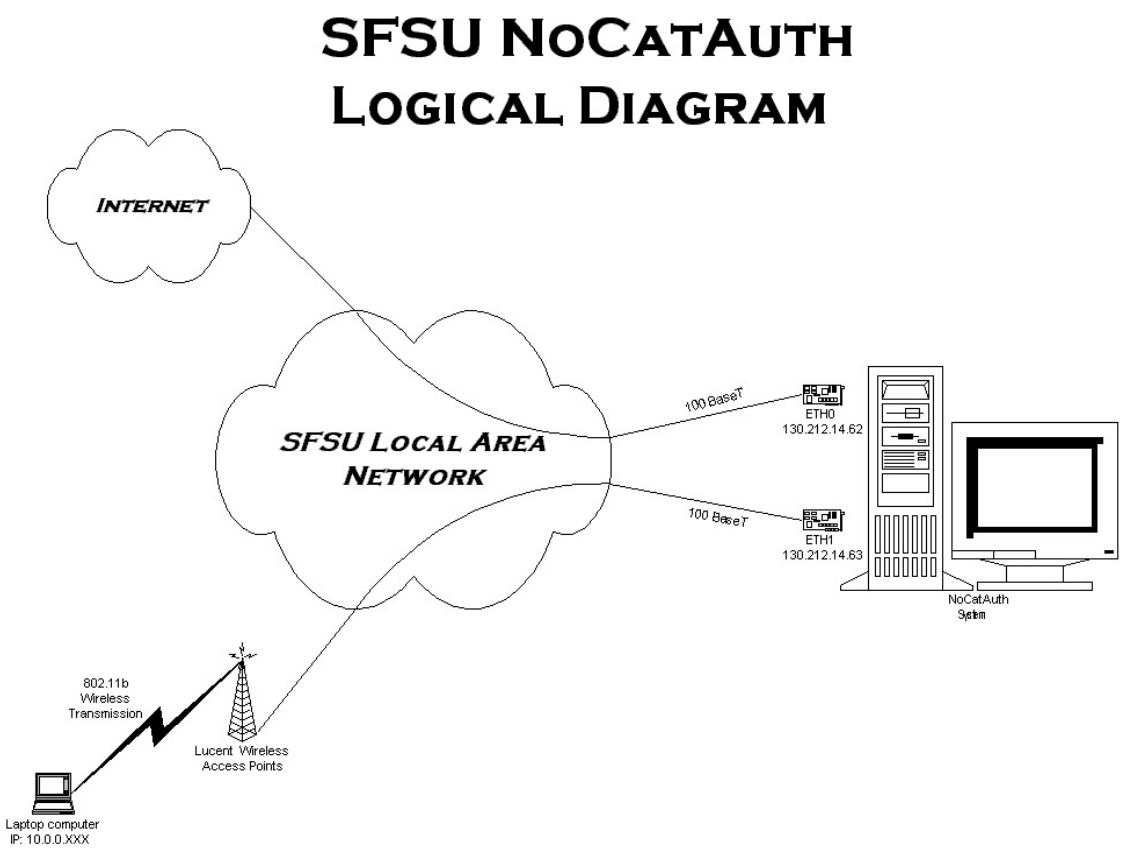
automated methods are used for creating classes, NoCatAuth chooses to create classes based on the role that a user plays.



The approach of using a "plug-in" file for providing a traffic shaper in NoCatAuth adds a lot of flexibility to the system. This leads us to believe that the implementation of class-based queues in NoCatAuth is simply the beginning of many other possibilities. NoCatAuth allows for a plug-and-play implementation of bandwidth throttling. By getting creative with its throttle script (throttle.fw) one could implement not just CBQ but many other Traffic Shaping methods such as stochastic fair queuing (SFQ). More detail on the various methods used for quality of service and related issues can be found in the following RFCs: 1349 (Almquist, 1992), 1812 (Baker, 1995), 2205 (Braden et al., 1997) and 2309 (Braden et al., 1998).

The netfilter rules are implemented via IPTables in the /bin/iptables/ folder. The initialize.fw script does most of the firewall initialization. It also does the initial marking. In our implementation, in order to prevent certain End-Users from consuming all the bandwidth, netfilter tools and IPTables were configured to limit bandwidth to 128k for most users. Owners of the NoCatAuth system were allowed the maximum amount of bandwidth.

H. Diagram


SFSU NoCatAuth
Logical Diagram
INTERNET
SFSU Local Area Network
100 BaseT
ETH0
130.212.14.62
100 BaseT
ETH1
130.212.14.63
NoCatAuth System
802.11b Wireless Transmission
Lucent Wireless Access Points
Laptop computer
IP: 10.0.0.XXX

Implementation

- For the project, an Intel Based PC was used. Below are the hardware specifications:

  - Disk Drives
    - Hard Drive
      - Western Digital WDC AC26400B
    - CD-Rom
      - Mitsumi FX3400
    - Floppy Disk
      - Standard
      - Iomega ZIP 100
  - Display
    - ATI Tech. Inc. 3D Rage Pro AGP 2X
  - Network Adaptors
    - Intel 8255X- based PCI Ethernet Adapter
    - Intel ® Pro/100 + PCI Adapter
  - Sound
    - Creative SB16 Compatible

- The Linux Distribution Package used was Red Hat Linux 7.1. The Install type selected was workstation. The Hard drive was manually partitioned, using the following partition table:

/	2 gig	hda5
/boot	20 Meg	hda1
/swap	917 Meg	hda10
/usr	2 gig	hda6
/home	500 Meg	hda7
/var	200 Meg	hda9
/tmp	500 Meg	hda8

- The Networking interfaces were configured as follows:

Eth0	
IP	130.212.14.62
Netmask	255.255.255.0
Network	130.212.14.0
Broadcast	130.212.14.255
Hostname	nocatout
Gateway	130.212.14.254
Primary DNS	130.212.10.163
Secondary DNS	130.212.10.162
Eth1	
IP	130.212.14.63
Netmask	255.255.255.0
Network	130.212.14.0
Broadcast	130.212.14.255
Hostname	nocatin
Gateway	130.212.14.254

Primary DNS	130.212.10.163
Secondary DNS	130.212.10.162

- No Firewall was selected. We chose not to install RedHat's firewalls because we weren't sure how it would work with NoCatAuth's implementation of Netfilter.

- A number of packages were chosen during installation, as follows:

  Databases
  - MySQL

  Internet
  - Open SSL-Perl

  System
  - Auth_ldap (for future use)
  - Gnome-linuxconf
  - Linuxconf

  Development
  - Languages
  - Perl
  - PHP (to support PHPMyAdmin)

  System Environment
  - IPTables v1.2.4
  - Wireless Tools

  Daemons
  - DHCP
  - OpenLDAP(for future use)
  - OpenSSH
  - HTTP + HTTPS

- In addition to the root account, two additional users accounts were created (pchang and hishii).

- GNU Privacy Guard was installed from www.gnupg.org/download.html.

- The IPRoute2 Package was installed from ftp.inr.ac.ru/ip-routing.

- PHPMyAdmin version 2.2.0 was downloaded and installed from www.phpwizard.net. PHPMyAdmin is a MySQL Database configuration tool.

  - The MySQL database information was populated using NoCatAuth's SQL script.

- The Apache web server was configured by editing the httpd.conf configuration file.

  - Apache was configured, and the pointers to Apache's cgi-bin were changed to point to /usr/local/nocat/cgi-bin.

  - MOD_SSL was obtained and installed from http://www.modssl.org/

  - Document root of Apache was pointed to /usr/local/nocat/htdocs.

- NoCatAuth **version 0.60** was downloaded from http://nocat.net/, and uncompressed into usr/local/nocat.

  - The command make gateway was issued, and the files were uncompressed into usr/local/nocat.

- The command make authserv was issued, to install and configure the authentication server
- The command make keys were made to create the PGP keys and certificates for the auth server.
- The NoCatAuth configuration file was edited as follows:
  - LocalNetwork_Eth1 – 130.212.14.63/255.255.255.0
  - DNSAddr – 130.212.10.163
  - Configured NoCatAuth to allow ports HTTP (80), FTP (20), and SSL (443) traffic
  - Modified AuthServiceAddr –130.212.10.62 – To use our own Authentication
  - Modified AuthServiceURL – https://130.212.14.62

A few problems arose during the implementation of the Secure Wireless Network Project. It took us a while to test it and make sure it works to some extent. One such problem was concerned with the trustedkeys.gpg file. After executing 'make pgpkey', trustedkeys.gpg was created by the NoCatAuth script into the NoCat default folder (/usr/local/nocat), which was not in the same directory as /usr/local/nocat/pgp/. To correct this problem we moved the trustedkeys.gpg to the /usr/local/nocat/pgp/ folder. Note that a stock trustedkeys.gpg came with the NoCatAuth script that works with NoCatAuth's auth server at auth.nocat.net. The purpose of this stock trustedkeys.gpg is to make a gateway work with the auth service at auth.nocat.net . This trustedkeys.gpg will not work if it is run in CAPTIVE MODE with a different AuthServ. When running in CAPTIVE MODE, the trustedkeys.gpg file that is generated by *your* auth server must be moved to the pgp folder.

Another problem occurred during the process of making the pgp keys. The system asked for a passphrase, which was not necessary[2] (Flickenger, 2001). Since the NoCatAuth script did not use a passphrase, the Gateway and AuthServ were not communicating because it expected a passphrase. To fix this issue, the passphrase was left blank during the process of 'make pgpkey'. Another problem is that the ownership was changed from root to apache for all the files under the nocat/pgp/ folder. This solved the permission issue between the gateway and auth server during communication.

Finally, in the AuthServ script 'AuthService.pm', the 4th line down from sub gateway_ip, the `return $gw;` was commented out. Without commenting this out, the login keeps looping. All this is with respect to version 0.60

[2]This information is now reflected in the NoCatAuth directions.

Results

Implementation of a secure wireless network works well, but does not protect from WEP-related attacks. In the notion of security, the only part that is left open is the encryption-over-air part. This is probably solvable by creating a VPN, where secure point-to-point tunnels or layer 2 tunnels are created. The discussion about the benefits of WEP vs. its vulnerability is outside the scope of this document.

Although traffic shaping is available based on the different types of users, bottlenecks at the gateway server may exist during peak times. If bottlenecks exist, it would be a good idea to only run the necessary services to keep the gateway server operating at peak efficiency.

The gateway and AuthServ can be used in a wired network environment as well. This implementation is often called a choke point server. This would prevent unauthorized users from getting access to the network. The network at SFSU is fully accessible in certain areas, where anyone can basically come in with a laptop and plug into the network. Hopefully, this will change in the near future.

Conclusion

The NoCatAuth gateway system worked as intended. The NoCatAuth code is still very beta, and much work needs to be done before it can be considered a complete product. It is the most promising software released to date. The system needs to be put through a stress test, before it is fully implemented.

The fact that NoCatAuth is available in the open source domain was a great advantage. We had access to the entire source code and were able to “look under the hood” when necessary. The open source community related to NoCatAuth (available via their mailing list) was also very insightful. Next steps include: Implementing a system to monitor and track usage of the system, a better interface for user configuration and management, and implementing the entire NoCatAuth system into a user friendly, all in one box.

References

Almquist, P. (1992). *RFC 1349: Type of Service in the Internet Protocol Suite*, [Web]. IETF. Available: http://www.ietf.org/rfc/rfc1349.txt

Armitage, G. (2000). *Quality of Service in IP networks: Foundations for a Multi-Service Internet* (1 ed.): Macmillan Technical Publishing.

Baker, F. (1995). *RFC 1812: Requirements for IP Version 4 Routers*, [Web]. IETF. Available: http://www.ietf.org/rfc/rfc1812.txt

Braden, B., et al. (1998). *RFC 2309:Recommendations on Queue Management and Congestion Avoidance in the Internet*, [Web]. IETF. Available: http://www.ietf.org/rfc/rfc2309.txt

Braden, R., Zhang, L., Berson, S., Herzog, S., & Jamin, S. (1997). *RFC 2205:Resource ReSerVation Protocol (RSVP)*, [Web]. IETF. Available: http://www.ietf.org/rfc/rfc2205.txt

Dierks, T., & Allen, C. (1999). *RFC 2246: The TLS Protocol*, [Web]. IETF. Available: http://www.ietf.org/rfc/rfc2246.txt

Fennelly, C. (2001). *Security in wireless*, [Web]. IBM Developer Works. Available: http://www-106.ibm.com/developerworks/library/wi-sec.html?dwzone=wireless

Fisher, D. (2001). *802.11 Wireless Security Holes Exposed*, [Web]. ExtremeTech. Available: http://www.extremetech.com/article/0,3396,s%253D201%2526a%253D11271,00.asp

Flickenger, R. (2001). How do I fix the passphrase problem? In a conversation with S. Verma. San Francisco, via AOL Instant Messenger.

Flickenger, R. (2002). *Building Community Wireless Networks* (1 ed.). Sebastopol, CA: O'Reilly & Associates.

Gomes, L. (2001). *Many wireless networks open to attack*, [Web]. ZDNet News. Available: http://zdnet.com.com/2100-11-529460.html?legacy=zdnn

Panko, R. (2001). *Business Data Communications and Networking* (3 ed.). New Jersey: Prentice Hall.

Radhakrishnan, S. (2001). *Linux - Advanced Networking Overview, Version 1*, [Web]. Available: http://qos.ittc.ukans.edu/howto/

Russell, P. (2001). *Netfilter: Firewalling, NAT and packet mangling for Linux 2.4*. Netfilter.org. Available: http://www.netfilter.org/documentation/index.html

Shipley, P. (2001). *Open WLANs: the early results of wardriving*, [Web]. dis.org. Available: http://www.dis.org/filez/openlans.pdf

Taylor, S., & Hettick, L. (2001). *QoS at the IP layer, Part 1*, [Web]. Network World Fusion. Available: http://www.nwfusion.com/newsletters/converg/2001/01123443.html

Verton, D. (2001). *Flaws in Wireless Security Detailed*, [Web]. Computerworld. Available: http://www.computerworld.com/cwi/stories/0,1199,NAV47-81_STO62220,00.html

Open Publication License

Draft v1.0, 8 June 1999

I. REQUIREMENTS ON BOTH UNMODIFIED AND MODIFIED VERSIONS

The Open Publication works may be reproduced and distributed in whole or in part, in any medium physical or electronic, provided that the terms of this license are adhered to, and that this license or an incorporation of it by reference (with any options elected by the author(s) and/or publisher) is displayed in the reproduction.

Proper form for an incorporation by reference is as follows:
Copyright (c) <year> by <author's name or designee>. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, vX.Y or later (the latest version is presently available at http://www.opencontent.org/openpub/). The reference must be immediately followed with any options elected by the author(s) and/or publisher of the document (see section VI).

Commercial redistribution of Open Publication-licensed material is permitted.

Any publication in standard (paper) book form shall require the citation of the original publisher and author. The publisher and author's names shall appear on all outer surfaces of the book. On all outer surfaces of the book the original publisher's name shall be as large as the title of the work and cited as possessive with respect to the title.

II. COPYRIGHT

The copyright to each Open Publication is owned by its author(s) or designee.

III. SCOPE OF LICENSE

The following license terms apply to all Open Publication works, unless otherwise explicitly stated in the document.

Mere aggregation of Open Publication works or a portion of an Open Publication work with other works or programs on the same media shall not cause this license to apply to those other works. The aggregate work shall contain a notice specifying the inclusion of the Open Publication material and appropriate copyright notice.

SEVERABILITY. If any part of this license is found to be unenforceable in any jurisdiction, the remaining portions of the license remain in force.

NO WARRANTY. Open Publication works are licensed and provided "as is" without warranty of any kind, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or a warranty of non-infringement.

IV. REQUIREMENTS ON MODIFIED WORKS

All modified versions of documents covered by this license, including translations, anthologies, compilations and partial documents, must meet the following requirements:

1. The modified version must be labeled as such.
2. The person making the modifications must be identified and the modifications dated.
3. Acknowledgement of the original author and publisher if applicable must be retained according to normal academic citation practices.
4. The location of the original unmodified document must be identified.
5. The original author's (or authors') name(s) may not be used to assert or imply endorsement of the resulting document without the original author's (or authors') permission.

V. GOOD-PRACTICE RECOMMENDATIONS

In addition to the requirements of this license, it is requested from and strongly recommended of redistributors that:

1. If you are distributing Open Publication works on hardcopy or CD-ROM, you provide email notification to the authors of your intent to redistribute at least thirty days before your manuscript or media freeze, to give the authors time to provide updated documents. This notification should describe modifications, if any, made to the document.
2. All substantive modifications (including deletions) be either clearly marked up in the document or else described in an attachment to the document.
3. Finally, while it is not mandatory under this license, it is considered good form to offer a free copy of any hardcopy and CD-ROM expression of an Open Publication-licensed work to its author(s).

VI. LICENSE OPTIONS

The author(s) and/or publisher of an Open Publication-licensed document may elect certain options by appending language to the reference to or copy of the license. These options are considered part of the license instance and must be included with the license (or its incorporation by reference) in derived works.

A. To prohibit distribution of substantively modified versions without the explicit permission of the author(s). "Substantive modification" is defined as a change to the semantic content of the document, and excludes mere changes in format or typographical corrections.

To accomplish this, add the phrase `Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.' to the license reference or copy.

B. To prohibit any publication of this work or derivative works in whole or in part in standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

To accomplish this, add the phrase 'Distribution of the work or derivative of the work in any standard (paper) book form is prohibited unless prior permission is obtained from the copyright holder.' to the license reference or copy.

OPEN PUBLICATION POLICY APPENDIX:

(This is not considered part of the license.)

Open Publication works are available in source format via the Open Publication home page at http://works.opencontent.org/.

Open Publication authors who want to include their own license on Open Publication works may do so, as long as their terms are not more restrictive than the Open Publication license.

If you have questions about the Open Publication License, please contact David Wiley, and/or the Open Publication Authors' List at opal@opencontent.org, via email.

To subscribe to the Open Publication Authors' List:
Send E-mail to opal-request@opencontent.org with the word "subscribe" in the body.

To post to the Open Publication Authors' List:
Send E-mail to opal@opencontent.org or simply reply to a previous post.

To unsubscribe from the Open Publication Authors' List:
Send E-mail to opal-request@opencontent.org with the word "unsubscribe" in the body.

Notes: