Project	**IEEE 802.20 Working Group on Mobile Broadband Wireless Access** <**http://grouper.ieee.org/groups/802/mbwa**>
Title	Wireless Security Threats
Date Submitted	**2003-01-12**
Source(s)	Alan Chickinsky Northrop Grumman/TASC 4801 Stonecroft Blvd Chantilly Va 20151 Voice: 703-633-8300 x8554 Fax: 703-449-4900 Email: achickinsky@northropgrumman.com
Re:	Security Tutorial
Abstract	Unlike wire based MANs, wireless MANs pose unique security problems. To monitor communications on a wire based MAN; the observer must be physically located near or in direct contact with the wire medium. In the wireless MAN, the observer must be within the broadcast signal range. Because one cannot completely control the signal broadcast pattern, there must be message encryption to limit interception. Encryption of a message requires a plain text message, an algorithm, and a key. To quickly and accurately decipher the encrypted message, one needs the algorithm, a key and the encrypted message. In an IEEE 802.20 Wireless MAN, an ease dropper has the encrypted message, and the algorithm. Therefore the IEEE 802.20 must select a method that makes it difficult for the ease dropper to discover the key. This paper proposes several methods to limit the ability to discover the key. It also proposes methods to discover data forgery of valid messages
Purpose	Address security issues for a Wireless MAN
Notice	This document has been prepared to assist IEEE 802.20 MBW. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.20 MBWA.
Patent Policy	The contributor is familiar with IEEE patent policy, as outlined in Section 6.3 of the IEEE-SA Standards Board Operations Manual <http://standards.ieee.org/guides/opman/sect6.html#6.3> and in *Understanding Patent Issues During IEEE Standards Development* <http://standards.ieee.org/board/pat/guide.html>.

Wireless Security Threats

Date: January 11, 2003

Author: Alan Chickinsky
Northrop Grumman/TASC
4801 Stonecroft Blvd
Chantilly, VA 20151
Phone: 703-633-8300 ext. 8554
Fax: 703-449-3400
e-Mail: achickinsky@Northropgrumman.com

Abstract

Unlike wire based MANs, wireless MANs pose unique security problems. To monitor communications on a wire based MAN; the observer must be physically located near or in direct contact with the wire medium. In the wireless MAN, the observer must be within the broadcast signal range. Because one cannot completely control the signal broadcast pattern, there must be message encryption to limit interception.

Encryption of a message requires a plain text message, an algorithm, and a key. To quickly and accurately decipher the encrypted message, one needs the algorithm, a key and the encrypted message. In an IEEE 802.20 Wireless MAN, an ease dropper has the encrypted message, and the algorithm. Therefore the IEEE 802.20 must select a method that makes it difficult for the ease dropper to discover the key.

This paper proposes several methods to limit the ability to discover the key. It also proposes methods to discover data forgery of valid messages.

Introduction

The IEEE 802.20 working group is taking a Systems Engineering approach to selecting a method that makes it difficult for the ease dropper to decrypt a message.

The first step in the process is to identify the types (or classes) of events that result in key discovery. The second step determines which events have a high probability that it will occur in a Wireless MAN. The third step lists candidate methods and describes how each minimizes the accurate message interception.

Security Events

This paper divides security events into Human Initiated Events, Data Privacy, Data Forgery, Denial of Service, and Hardware Error classes.

Human Initiated Events

Most security events occur due to human errors. Two common errors are administrators improperly configure systems and system software errors. Hackers have exploited these errors to gain access to the messages. There is nothing that a specification can do to avoid or eliminate these errors.

Data Privacy

A Data Privacy security event occurs when an unauthorized user gains access to all the unencrypted traffic between two points. To decrypt the message, the user needs the encrypted message, the algorthym and the key. Since the wireless LAN Specification defines the algorthym, and the user has the encrypted message, then only the messagef key is missing. A hacker can discover the key if one is given the encrypted text, the algorithm and the original plain text message. This approach assumes that the only the original text is encrypted and sent. No additional noise or filler characters are added to the encrypted text.

There are a number of ways for a hacker to access the plain text messages. In a wireless environment, traffic leaving a wireless user is encrypted at the MAC level by the wireless interface. The encrypted data is then sent to an access point, where it is decrypted. The access point sends the plain text data to the router. If the Hacker could modify the routing information, it would be possible for the plain text message to be sent to a compromised system for recording. The manner used to intercept the plain text message will not help establish the required encryption algorithm.

Data Forgery

A Data Forgery security event occurs when an unauthorized user inserts data into network as a valid user. There are two ways valid data enters the network, replay and mimicking.

To replay a message, data on the network is recorded, and then replayed at a later time. The replay could be either an exact copy, or a modified transmission. The modified message could route the decrypted message to a comprised site.
To create a modified message, the hacker needs a limited knowledge of the plain text message, some knowledge of the message format, and good method to replay a message. This type of attack is used to discovery clear text messages. An exact copy attaché, starts with the hacker recording the traffic to/from a specific station, then at a later time replay the traffic. A replay attack is useful when a privileged user grants temporary rights on a user's station. The Hacker would replay the message traffic, and the terminal would ganted temporary rights, without the knowledge of the privileged user.

The second type of forgery attack the hacker attempts to mimic a valid station on the network. This attack starts when the hacker discovers the valid key. Once the key is discovered and used, the receiving station would not be able to distinguish between the two transmitting stations.

Denial of Service

This occurs when the intruder floods the network with either valid or invalid messages. The intruder could jam the channel, inhibiting all transfer of information. The intruder could also flood a receiving station with valid garbage messages. These messages would force the system to use valuable battery power. When the power is exhausted, a denial of service is accomplished. There is nothing that a specification can do to avoid or eliminate these errors.

Hardware errors

When a hardware error occurs, the hardware could cease to function, jam the channel by sending constant, or send random patterns on the channel. This random pattern may be received as a valid message for another station. The result of a random pattern is undetermined at this time. These types of errors are detected and prevented at the PHY layer.

Possible Solutions at the MAC Layer

For Data Privacy –

The following paragraphs describe how a hacker can get (1) the first part of an encrypted message (2) the first part of a plain text message and (3) Encryption algorithm.

Lets assume the MAC level encrypts only the message payload. For most current implementations, message payloads at the MAC level start with an IP header. RFC 791, section 3.1 defines the header format for each IP message (see Appendix A). The first 128 bits contain only three fields that normally are changed, "Identification", "Time to Live", "Message Size" and "Checksum". The "identification" is a sequence nuimber that is fixed for each message fragment. Since a sequence number is used, and it is most likely repeated, it is easy to guess. The "Message Size" is the number of characters in the encrypted message. Assuming an implementation uses a constant for the "Time to Live", a hacker can compute the "Checksum". Thus in theory, the hacker has the first 128 bits of clear text message.

By definition, the hacker has intercepted the encrypted message and has the IEEE 802.20 specification that defines the encryption algorithm.

The following paragraphs show how a hacker, given the information described above, could discover the current key.

Most encryption algorithms use the following formulas.

E (i) = k(i) <operator> p(i) for i = 1 to number of bits in the key

Where
E(i) is the i-th bit of the Encrypted message
k(i) is the i-th bit of the key
<Operator> is the mathematical operator
p(i) is the i-th bit of the plain text operator

The developers of encryption decided increase security by adding a non-repetitive element, giving-

E (i) = k(i) <operator> p(i) <operator> m(i) for i = 1 to number of bits in the key

Where
E(i) is the i-th bit of the Encrypted message
k(i) is the i-th bit of the key
<operator> is the mathematical operator
p(i) is the i-th bit of the plain text operator
m(i) is the i-th bit of the message sequence key

The developers of encryption decided increase security by creating a mathematical function that uses all key bits and plain text data bits in a complex set of functions for each encrypted bit, giving-

E (i) = f {k, p, m} for i = 1 to number of bits in the key

Where
E(i) is the i-th bit of the Encrypted message
k is the key
f {…} is the mathematical function
p is the plain text operator
m is the message sequence key

Again, given the first part of two encrypted messages, the first part of two plain text messages, and the relationship between sequence numbers, solving for the key is a simple task.

If we change the encryption algorithm to randomly add bytes to the plain text message we now have an equation that always has a random pattern. This produces a different equation for each message, and creates an infinite number of equations with an infinite number of variables. By prefixing each message with a fixed number of random bits, the key is randomised for each transmission. The prefix bits only appear on the wireless transmission. The number of random bits should equal the key size in a block structured encryption algorithm.

Additional complexity can be created by randomly adding a character sequence to the message. The sequence would have an escape character followed by a random character. This sequence would be ignored when decrypting a message. Since all character sequences are possible in a message, the two-character sequence of escape character and escape character is replaced by one escape character.

This type of additional complexity can be compromised if the Network layer always sends maximum IP block size messages. Without a major impact on network performance, the Network layer could define a maximum message size as a random number between maximum IP packet size – 20 and maximum IP packet size – 5. This means all messages sent will contain a minimum of 0 to 10 filler character message sequences. As an alternative, the specification could define that all messages are sent as one of x specified block sizes. Messages with one or more filler character message sequences will be padded till the block size if achieved.

Randomly adding byte sequences to a message and fixed block size changes the number of plain text messages sent. With varying fixed message sizes and number of messages, the Hacker is unable to reliably guess when a known plain text message is transmitted.

The above discussion assumes that a new pseudo key is created for each block of "n" characters. The pseudo key is based on the previous encryption results.

Assigning the first key

There is an issue at startup, how is a secure channel created? You will recall from the previous presentation, a secure channel requires 4 elements, the plain text message, the encrypted message, the encryption algorithm and the key. Of the four elements, the bad guy knows the encrypted message, and the algorithm. Assuming the bad guy does not know the plain text message, then what key do we use?

Considering we are developing a "public" communications system, do we announce a public key for all to use? Now the bad guy has the key. Do we download the key on a clear channel? If we do, again the bad guy can read the clear channel.

What we need is a way to get the first key on a different media than the radio channel.

Guessing the key

Lets assume that a bad guy can insert a message into the data stream. The bad guy knows the format of the response. Now the bad guy sends a "null" message. So he gets a canned response properly encrypted. Next, the bad guy sends a second message. Again he receives a canned response. But the second response differs from the first by a small number of bits. Using these two messages, the bad guy can quickly guess the key.

The above example assumes the sender uses the same key. To avoid this attack, each message is sent with a different key. There are two ways to assure each message is sent with a different key.

Different keys

In the first method, each station maintains a list of keys to be used. Each message is encoded with one key. After the key is used, it is destroyed. When the list of keys is exhausted, a new list is created.

The second method creates a key with a sequence number and a fixed part. Each message is sent with a number generated in sequential order. This number is called the initialization vector (IV). Appending the IV to the fixed key, a unique encryption key is created, and every message is encrypted with a unique key.

The idea of never reusing a key, avoids any problems of recording a message stream then replaying the stream to gain access.

Distributing the keys

We have already addressed the topic of initial key creation. For this discussion, we will assume there are secure channels between the mobile or base station and the key distribution server. One the keys are exhausted (either the list or the sequence number will roll over) the station contacts the key server for a new key. The protocol used is defined in 802.1x. 802.11i has noticed some issues when implementing this protocol. They are currently in discussions with 802.1x to resolve the issue.

Key, key, who has the key?

Lets assume we have a station traveling at 250 km/hr. This means the station could send one message in cell 1 and receive the response in cell 2. Remember to accept a packet, the base station needs the correct key. If we are using an IV, then the second base station needs the "next" sequence number. For this case, we could transfer all keys to the adjacent cells. But what happens if the station sends a message in cell 1, passes through cell 2 and sends a retransmission in cell 3. From a security point of view, if we have a bad guy in cell 3, then we could have a valid message sent in cell 1. Then the bad guy sends a mimic in cell 3. How does the system tell the difference?

For Data Forgery

Data forgery has two different attacks, replay and mimicking.

Replay

If each message sent over the wireless link has a unique message number, and there is a fixed relationship between message numbers, then a replay attack would send messages with an incorrect message number. Once a station gets a message with an incorrect message number, the station must reject the message. A problem occurs when the message number is allowed to repeat during any given session. There is a small possibility that a recorded message would have the proper message number. To avoid this issue, the message number is part of encrypted message. When all the available message numbers are used, a new encryption key is required before further data message transfer can occur.

For this solution, the message number is a combination of a message number and key. Making the probability of message number repetition almost zero.

Mimicking

For this discussion, we assume (1) the Hacker does not have the valid key for the current session and (2) the distribution of keys and validation of any key requests are outside of this discussion. If one assumes that the approach under the data privacy section is adopted, then probability for discovery of a valid key is almost zero.

But, there is an issue on what does a system do when it "logs-off" the network. It is assumed that the key would be retired on termination. But this discussion is left for a discussion on key distribution.

Possible Solutions at the PHY Layer

For Data Forgery

In the MAC layer discussion, there were a number of conditions to do an adequate determination of Data Forgery. If we move the detection of a Data Forgery event to the PHY layer a better detection method is available.

In the PHY layer, the RF interface could use the subtle hardware differences in each transmitter to identify any type of data forgery. The cellular industry has been researching this issue. Using their discoveries, it should be possible to detect a Data Forgery event.

For the Hardware Errors

There are numerous algorithms and methods to detect errors and minimize hardware errors. Methods for this work are left for further discussion.

Conclusion

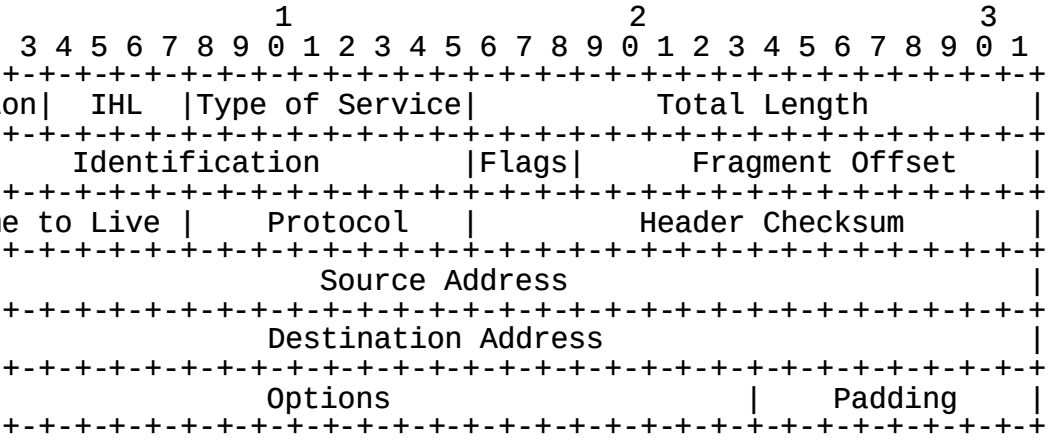
This paper has proposed several alternatives to create a secure environment to send data. More investigation is needed to determine if some subset of the proposed methods produce a secure environment.

Appendix A

Excerpt from RFC 791
"Internet Protocol" dated, September 1981
Pages 10 through 13

3.1. Internet Header Format

A summary of the contents of the internet header follows:



Example Internet Datagram Header

Figure 4.

Note that each tick mark represents one bit position.

Version: 4 bits

The Version field indicates the format of the internet header. This document describes version 4.

IHL: 4 bits

Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

Type of Service: 8 bits

The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network. Several networks offer service precedence, which somehow treats high precedence traffic as more important than other traffic (generally by accepting only traffic above a certain precedence at time of high load). The major choice is a three way tradeoff between low-delay, high-reliability, and high-throughput.

Bits 0-2: Precedence.
Bit 3: 0 = Normal Delay, 1 = Low Delay.

```
Bits   4:  0 = Normal Throughput, 1 = High Throughput.
Bits   5:  0 = Normal Relibility, 1 = High Relibility.
Bit  6-7:  Reserved for Future Use.

   0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+-----+
|                 |     |     |     |     |     |
|   PRECEDENCE    |  D  |  T  |  R  |  0  |  0  |
|                 |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+

  Precedence

    111 - Network Control
    110 - Internetwork Control
    101 - CRITIC/ECP
    100 - Flash Override
    011 - Flash
    010 - Immediate
    001 - Priority
    000 - Routine
```

The use of the Delay, Throughput, and Reliability indications may increase the cost (in some sense) of the service. In many networks better performance for one of these parameters is coupled with worse performance on another. Except for very unusual cases at most two of these three indications should be set.

The type of service is used to specify the treatment of the datagram during its transmission through the internet system. Example mappings of the internet type of service to the actual service provided on networks such as AUTODIN II, ARPANET, SATNET, and PRNET is given in "Service Mappings".

The Network Control precedence designation is intended to be used within a network only. The actual use and control of that designation is up to each network. The Internetwork Control designation is intended for use by gateway control originators only. If the actual use of these precedence designations is of concern to a particular network, it is the responsibility of that network to control the access to, and use of, those precedence designations.

```
  Total Length:  16 bits
```

Total Length is the length of the datagram, measured in octets, including internet header and data. This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole or in fragments). It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams.

The number 576 is selected to allow a reasonable sized data block to be transmitted in addition to the required header information. For example, this size allows a data block of 512 octets plus 64 header octets to fit in a datagram. The maximal internet header is 60 octets, and a typical internet header is 20 octets, allowing a margin for headers of higher level protocols.

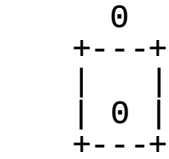
Identification: 16 bits

An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags: 3 bits

Various Control Flags.

Bit 0: reserved, must be zero
Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.
Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.



Fragment Offset: 13 bits

This field indicates where in the datagram this fragment belongs.

The fragment offset is measured in units of 8 octets (64 bits). Thefirst fragment has offset zero.

Time to Live: 8 bits

This field indicates the maximum time the datagram is allowed to remain in the internet system. If this field contains the value zero, then the datagram must be destroyed. This field is modified in internet header processing. The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Protocol: 8 bits

This field indicates the next level protocol used in the data portion of the internet datagram. The values for various protocols are specified in "Assigned Numbers".

Header Checksum: 16 bits

A checksum on the header only. Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed.

The checksum algorithm is:

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

This is a simple to compute checksum and experimental evidence indicates it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Source Address: 32 bits

The source address.