CRYPTOGRAPHY USING ARTIFICIAL NEURAL NETWORKS

EGEE 529

Submitted by: Abhishek Gaikwad
CWID: 893451187

1]Abstract:

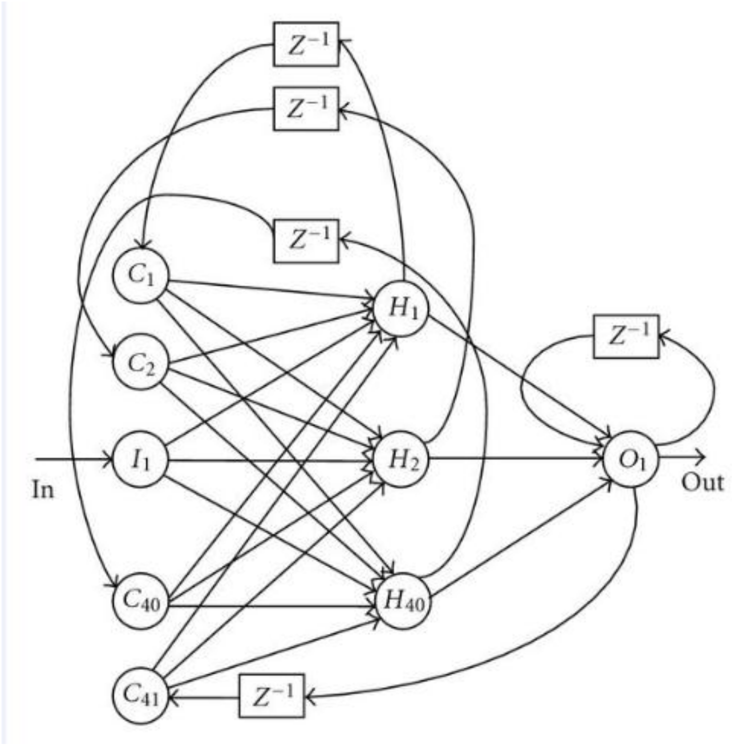
As discussed in the initial two presentations in the class, the document explains the implementation of cryptography using artificial neural networks. The documents explains us in depth integration of the model and how this model is implanted to a network security subnet.

2]Design of Artificial Neural Networks:

The artificial neural network to implement cryptography. As discussed earlier a Jordan neural network implementing backpropagation algorithm is designed. The Jordan neural network was implemented using a sequential machine.

Implementation of a Jordan Network:

The Jordan network to implement cryptography is developed by preparing a state diagram, obtained by the adder and detector combined action through a combinational circuit.



[1]Jordan Neural Net

3]Working of the Jordan network:

Here the O/P obtained is fed to the input through extra set of inputs called state inputs. The number of state inputs are equal to the number of outputs generated. The weight in between the output and input is fixed at +1. Network learns through the hidden layers.

In order to adjust weight in between the connections the following equations are used:

$\Delta_l a_s k=v\Delta_k y_j$

The above equation is generally the gradient decent to obtain the appropriate weight changes.

Now through the operation of the adder and detector, the output obtained is recorded in a state table. The input fed to the state table can be considered as the string to be encrypted and the output generated is treated as the decrypted information.

The training of the neural net is done through this sequential table. As the output set can be considered as a training set for reference. The output here is dependent on the starting state and the starting state is implemented as the public key to the encryption. The private key can be considered as the next table data in the sequential table.

Adder implementation

A serial adder is implemented as follows:

Step 1: Two input streams are defined at once.

Step 2:String length of the bits is defined

Step 3: The bits are compared to an arbitrary value s, bigger than the value

Step 4: if s<0!1 false, exit the sequence,else if s>0!1 .carry the operation

Step 5: The operation is add bit 1 and 2

Input bits can be generated by sequential generators or shift registers.

Sequence detector: The output obtained through a output is fed to a sequence detector. The sequence detector sole job is to produce a output 1 for all relevant equivalent input streams. The output if synchronization is not there, then output is generated as zero.

Thus if we correlate this and implement the Jordan network, the adder can be used to implent the simple input and hidden layers and state inputs. Whereas the output and state transition weights are implemented using a sequence detector.

4]Program structure:

Sequential machine implemented had 2 states 0&1 and input are implemented through a 3 bit string to be decrypted. For 0 state, the input word will be moved by 1 to develop the encrypted word ,while if for state i1, the letter is moved by 2.Thus state is automatically shifted. For instance if the input is taken as A, the outcome is B and there is a change in state to 1. If the state is again A the output will be the next word as input+1.

Outputs for the implemented logic:

Enter the states:4

Enter admin: AGGH

Password: CHHB

5]Application of logic to Network Security.

Now using a video management system system such as Genetec can implement the import of a matlab file to a hypertext markup file (XML file). The XML file consist of all the features of the camera, defined by us. Right from resolution to the authentication of the camera and VMS can be modified using the XML text.

Methodology

The ANN will allow the user to set a particular admin name to the stream. Then the algorithm will decode only the password (that is the decrypted output) for the particular stream and display the images. Thus to summarize the Ann will give out a password only for dedicated admin name, based on the state. The state is basically the stream number. From 1-4 we select the the state, and assign the username to a dedicated stream only.

Here we are considered with the authentication only, since we are here concerned with the cryptography of the system.

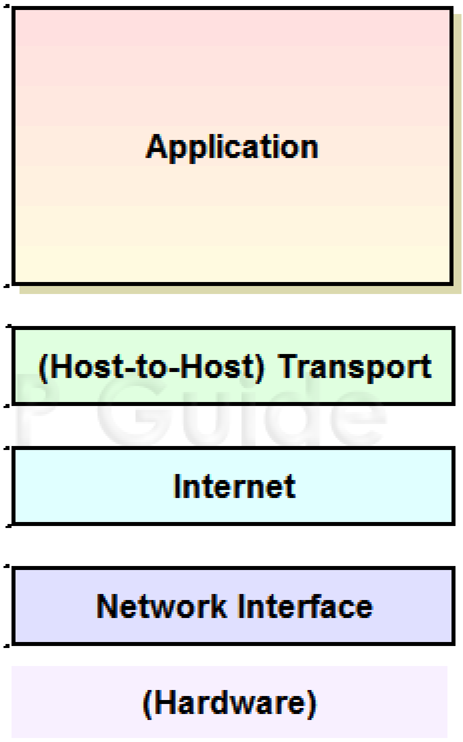
Importing matlab features to the XML file

The entire functionality of an .m file can be implemented to the model in the following way:

```
  <!-- Misc integration notes and FAQ:start Faq on ann.m import>

<import.ann.m/path/C:\Users\mega05\Desktop> // Locating the .m file on the system//

<typefile:.m>//Routine to import the .m file.//

<library:extern.msupport>//allowing support of .m file to the xml parameters//

<stringappend:pecification Name="inputencrypt" Value="true"appendendl.annm/> /routine to import the encrypt feature//

        <Specification Name="outputdecrypt" Value="true"appendendl.annmDeviceSpecs Product="1145" EntityType="Unit" authentication:append">//routine to import the decrypt feature//

        <Specification Name="inputencrypt" Value="true"appendendl.annm/>

        <Specification Name="outputdecrypt" Value="true"appendendl.annm/>

<stringappend:pecification Name="inputallspecifications" Value="true"appendendl.annm/>

        <Specification Name="outputspecifications" Value="true"appendendl.annm>

DeviceSpecs Product="1145" EntityType="Unit" authentication:append">

        <Specification Name="inputencrypt" Value="true"appendendl.annm/>

        <Specification Name="outputdecrypt" Value="true"appendendl.annm/>//

<!-- Misc integration notes and FAQ:end Faq on ann.m import>
```

Now the above sequence can be implemented on other features defined for the camera. But since cryptography is our scope, we will only implement the authentication feature.

6]Understanding the network to be implemented



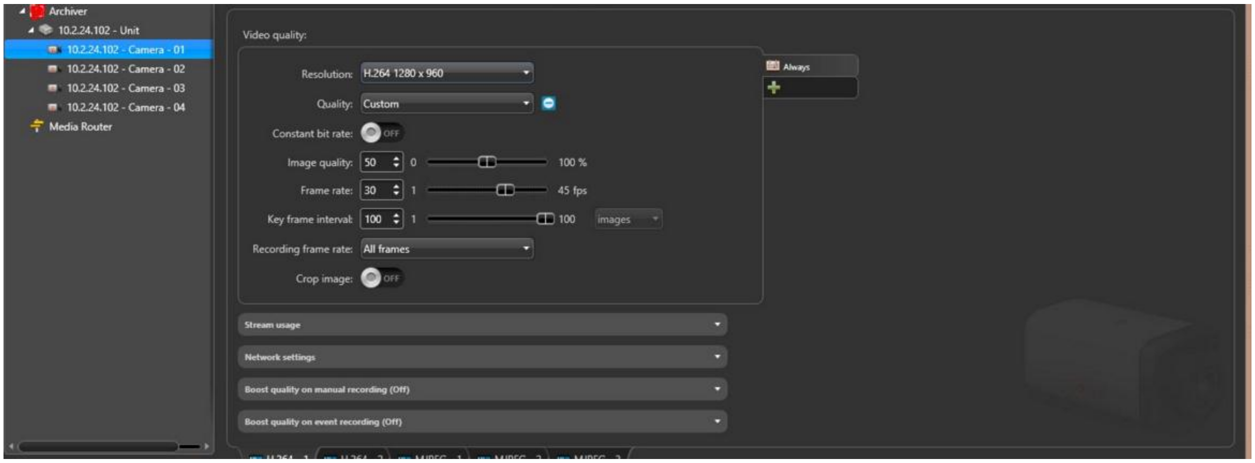
TCP/IP Model

As seen in the above model the ANN designed will be implemented between the internet and application layer. Thus doing this will help in slicing the H.264 frame to 3 components(that is ANN model will split the frame into 3 slices) i.e the I,P and N frame. Further the ANN model will extract the effective H.264 features fro the frames to develop a single frame, consisting of superior H.264 parameters to give us a QOS of 5. The protocols used here are RTSP/RTP.

7] Results of integration

First we have to give in the particular authentication to a particular camera stream for integration. The authentication feature will now use the modified XML with the .m path.

Thus when you try to implement a single authentication to other streams, the streams will immediately disconnect and the alarm will be triggered.


Archiver
10.2.24.102 - Unit
10.2.24.102 - Camera - 01
10.2.24.102 - Camera - 02
10.2.24.102 - Camera - 03
10.2.24.102 - Camera - 04
Media Router
Always
Stream usage
Network settings
Boost quality on manual recording (Off)
Boost quality on event recording (Off)

Video quality:

Resolution: H.264 1280 x 960

Quality: Custom

Constant bit rate: OFF

Image quality: 50 0 100 %

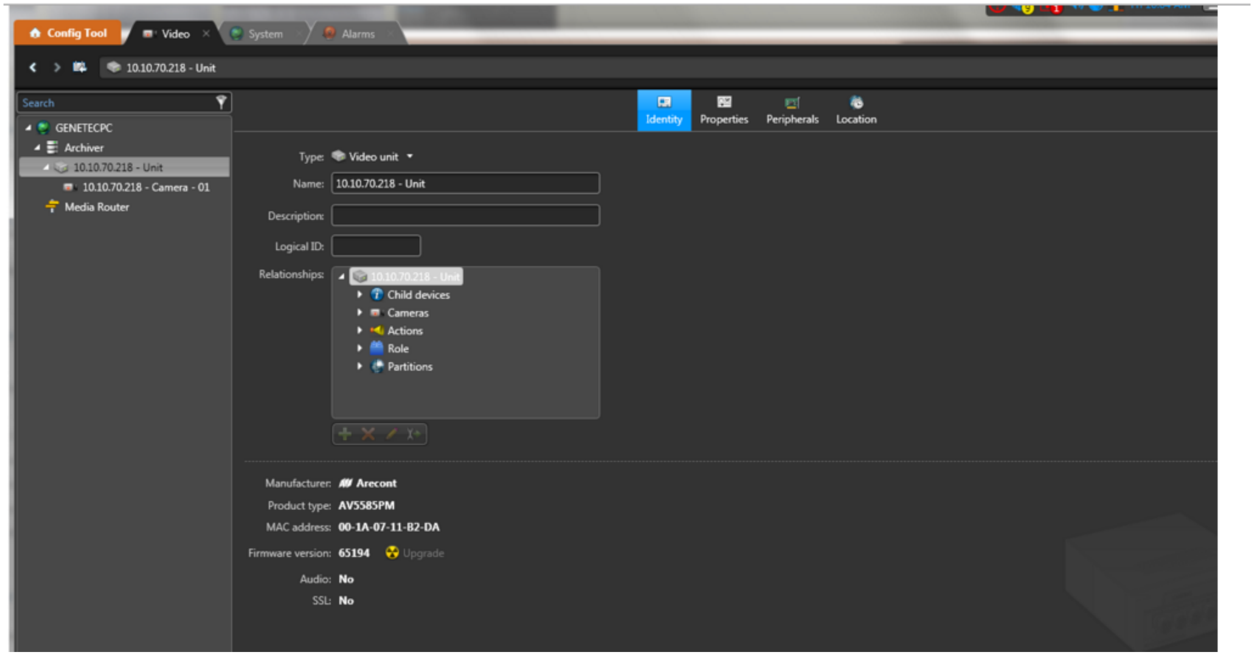
Frame rate: 30 1 45 fps

Key frame interval: 100 1 100 images

Recording frame rate: All frames

Crop image: OFF

4 Streams on not doing cross authentication implementation



Type: Video unit

Name: 10.10.70.218 - Unit

Description:

Logical ID:

Relationships:

Manufacturer: Arecont

Product type: AV5585PM

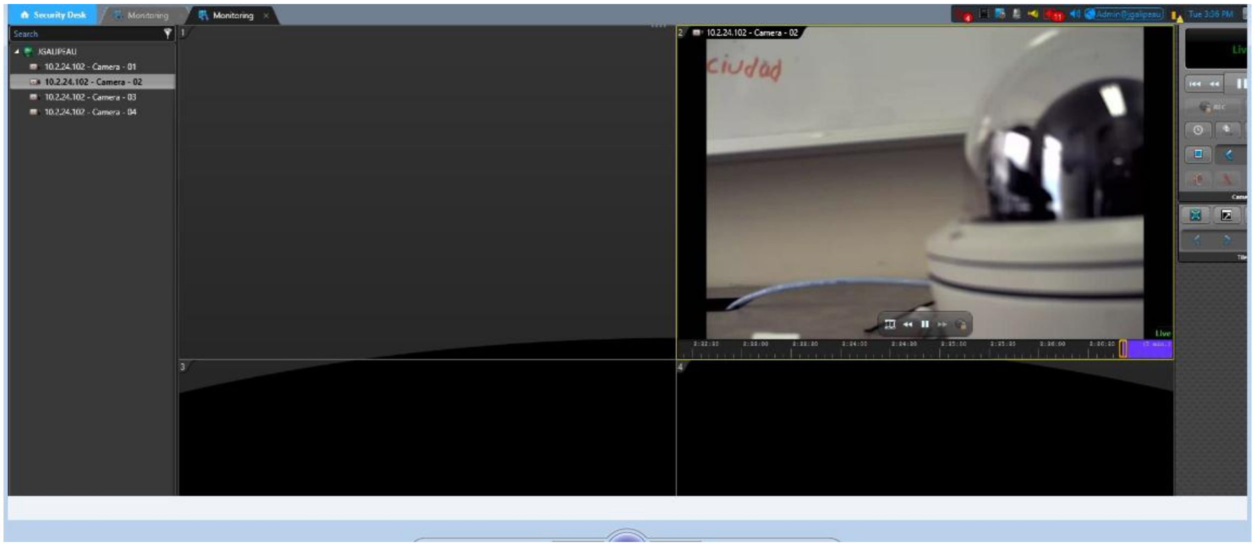
MAC address: 00-1A-07-11-B2-DA

Firmware version: 65194

Audio: No

SSL: No

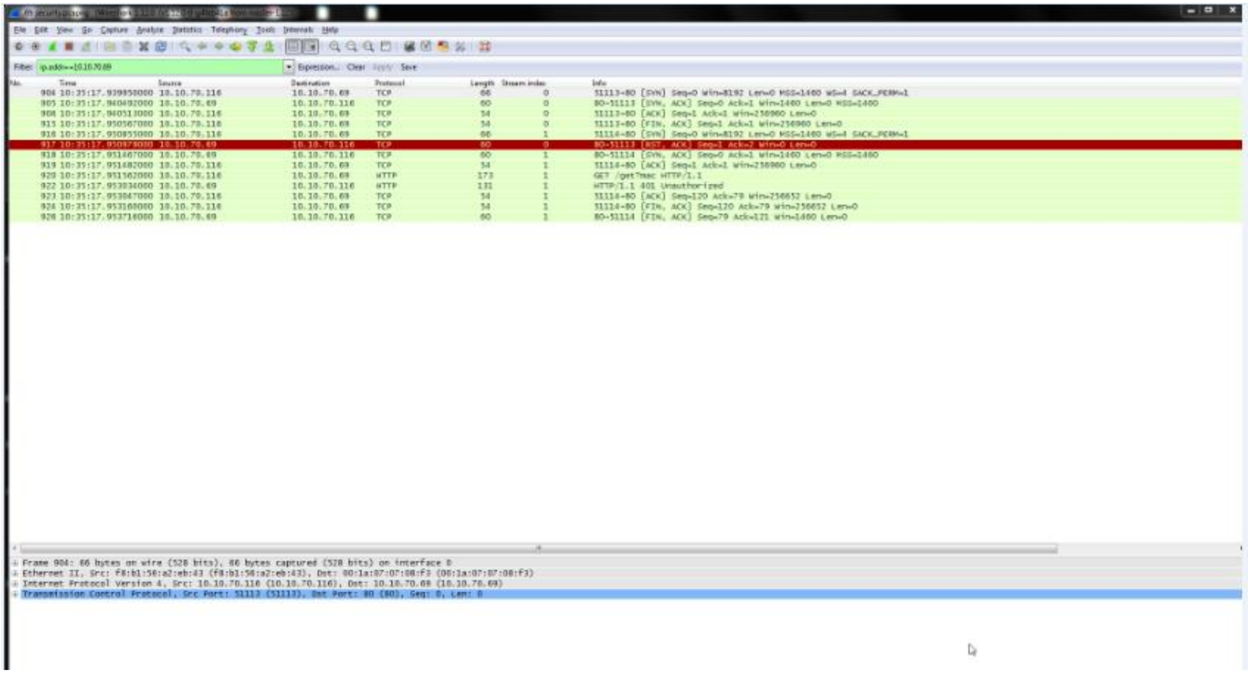
Streams getting disconnected on implementing the cross authentication


Security Desk
Monitoring
Monitoring
Search
JGALIPEAU
10.2.24.102 - Camera - 01
10.2.24.102 - Camera - 02
10.2.24.102 - Camera - 03
10.2.24.102 - Camera - 04
10.2.24.102 - Camera - 02
Live
ciudad

Client screen implementing disconnections and alarms triggered.

As seen above the alarms in red shows, intrusion and thus disconnects the other 3 streams.

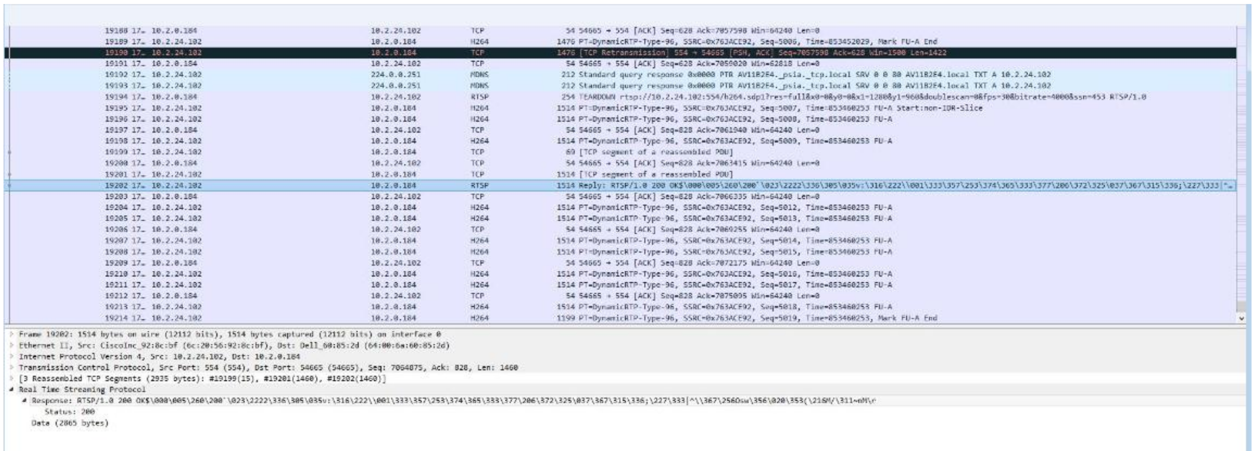
The wireshark capture below shows how the stream prevents authentication, by neglecting the intrusion mac ip.



```
Filter: ip.addr==10.10.70.69
No.  Time               Source        Destination   Protocol  Length  Stream index  Info
904  10:35:17.939950000 10.10.70.116  10.10.70.69   TCP       66      0             51113→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
905  10:35:17.940492000 10.10.70.69   10.10.70.116  TCP       60      0             80→51113 [SYN, ACK] Seq=0 Ack=1 Win=1460 Len=0 MSS=1460
908  10:35:17.940513000 10.10.70.116  10.10.70.69   TCP       54      0             51113→80 [ACK] Seq=1 Ack=1 Win=258900 Len=0
915  10:35:17.950567000 10.10.70.116  10.10.70.69   TCP       54      0             51113→80 [FIN, ACK] Seq=1 Ack=1 Win=258900 Len=0
916  10:35:17.950855000 10.10.70.116  10.10.70.69   TCP       66      1             51114→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
917  10:35:17.950979000 10.10.70.69   10.10.70.116  TCP       60      0             80→51113 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
918  10:35:17.951467000 10.10.70.69   10.10.70.116  TCP       60      1             80→51114 [SYN, ACK] Seq=0 Ack=1 Win=1460 Len=0 MSS=1460
919  10:35:17.951482000 10.10.70.116  10.10.70.69   TCP       54      1             51114→80 [ACK] Seq=1 Ack=1 Win=258900 Len=0
920  10:35:17.951562000 10.10.70.116  10.10.70.69   HTTP      173     1             GET /getTmac HTTP/1.1
922  10:35:17.953034000 10.10.70.69   10.10.70.116  HTTP      131     1             HTTP/1.1 401 Unauthorized
923  10:35:17.953047000 10.10.70.116  10.10.70.69   TCP       54      1             51114→80 [ACK] Seq=120 Ack=79 Win=258852 Len=0
924  10:35:17.953160000 10.10.70.116  10.10.70.69   TCP       54      1             51114→80 [FIN, ACK] Seq=120 Ack=79 Win=258852 Len=0
926  10:35:17.953716000 10.10.70.69   10.10.70.116  TCP       60      1             80→51114 [FIN, ACK] Seq=79 Ack=121 Win=1460 Len=0

+ Frame 904: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
+ Ethernet II, Src: f8:b1:56:a2:eb:43 (f8:b1:56:a2:eb:43), Dst: 00:1a:07:07:08:f3 (00:1a:07:07:08:f3)
+ Internet Protocol Version 4, Src: 10.10.70.116 (10.10.70.116), Dst: 10.10.70.69 (10.10.70.69)
+ Transmission Control Protocol, Src Port: 51113 (51113), Dst Port: 80 (80), Seq: 0, Len: 0
```

The highlighted red text shows us , the disconnection of the streams and hence successful implementation of our imported algorithm.

To verify this we ,see below the working of the system under normal conditions.

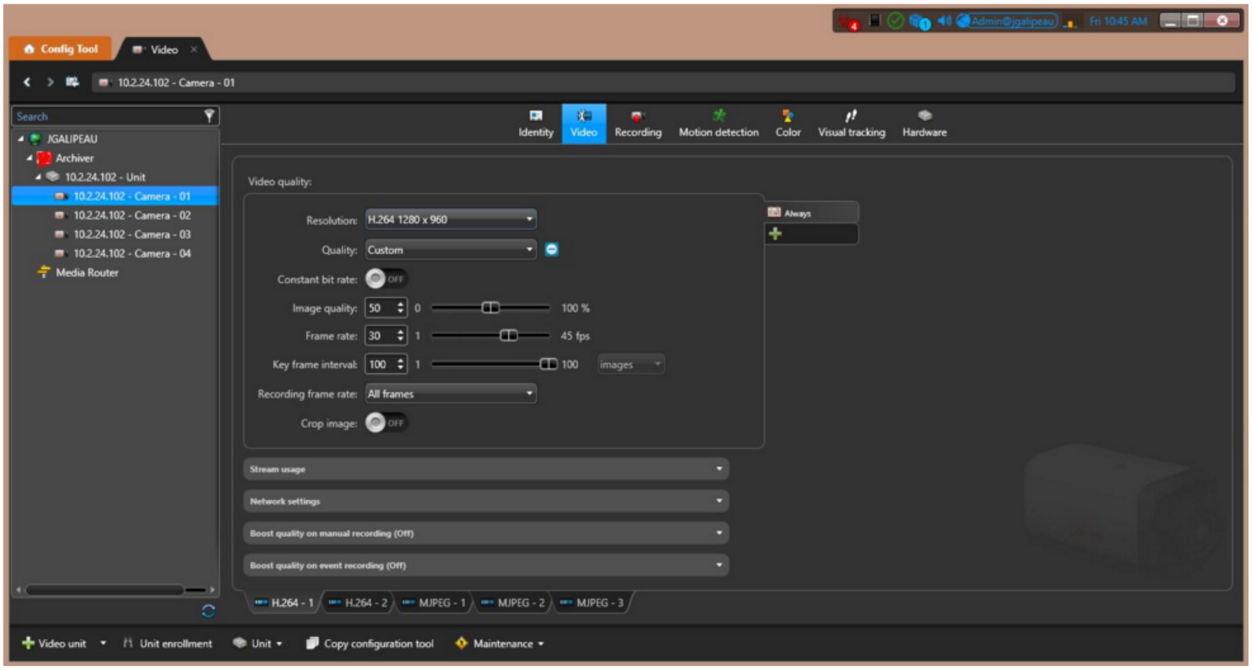


```
19188 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=628 Ack=7057598 Win=64240 Len=0
19189 17… 10.2.24.102    10.2.0.184   H264  1476 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5006, Time=853452029, Mark FU-A End
19190 17… 10.2.24.102    10.2.0.184   TCP   1476 [TCP Retransmission] 554 → 54665 [PSH, ACK] Seq=7057598 Ack=628 Win=1500 Len=1422
19191 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=628 Ack=7059020 Win=62818 Len=0
19192 17… 10.2.24.102    224.0.0.251  MDNS   212 Standard query response 0x0000 PTR AV11B2E4._psia._tcp.local SRV 0 0 80 AV11B2E4.local TXT A 10.2.24.102
19193 17… 10.2.24.102    224.0.0.251  MDNS   212 Standard query response 0x0000 PTR AV11B2E4._psia._tcp.local SRV 0 0 80 AV11B2E4.local TXT A 10.2.24.102
19194 17… 10.2.0.184     10.2.24.102  RTSP   254 TEARDOWN rtsp://10.2.24.102:554/h264.sdp1?res=full&x0=0&y0=0&x1=1280&y1=960&doublescan=0&fps=30&bitrate=4000&ssn=453 RTSP/1.0
19195 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5007, Time=853460253 FU-A Start:non-IDR-Slice
19196 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5008, Time=853460253 FU-A
19197 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=828 Ack=7061940 Win=64240 Len=0
19198 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5009, Time=853460253 FU-A
19199 17… 10.2.24.102    10.2.0.184   TCP     69 [TCP segment of a reassembled PDU]
19200 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=828 Ack=7063415 Win=64240 Len=0
19201 17… 10.2.24.102    10.2.0.184   TCP   1514 [TCP segment of a reassembled PDU]
19202 17… 10.2.24.102    10.2.0.184   RTSP  1514 Reply: RTSP/1.0 200 OK$\000\005\260\200`\023\2222\336\305\035v:\316\222\\001\333\357\253\374\365\333\377\206\372\325\037\367\315\336;\227\333|^…
19203 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=828 Ack=7066335 Win=64240 Len=0
19204 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5012, Time=853460253 FU-A
19205 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5013, Time=853460253 FU-A
19206 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=828 Ack=7069255 Win=64240 Len=0
19207 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5014, Time=853460253 FU-A
19208 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5015, Time=853460253 FU-A
19209 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=828 Ack=7072175 Win=64240 Len=0
19210 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5016, Time=853460253 FU-A
19211 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5017, Time=853460253 FU-A
19212 17… 10.2.0.184     10.2.24.102  TCP     54 54665 → 554 [ACK] Seq=828 Ack=7075095 Win=64240 Len=0
19213 17… 10.2.24.102    10.2.0.184   H264  1514 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5018, Time=853460253 FU-A
19214 17… 10.2.24.102    10.2.0.184   H264  1199 PT=DynamicRTP-Type-96, SSRC=0x763ACE92, Seq=5019, Time=853460253, Mark FU-A End

> Frame 19202: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
> Ethernet II, Src: CiscoInc_92:8c:bf (6c:20:56:92:8c:bf), Dst: Dell_60:85:2d (64:00:6a:60:85:2d)
> Internet Protocol Version 4, Src: 10.2.24.102, Dst: 10.2.0.184
> Transmission Control Protocol, Src Port: 554 (554), Dst Port: 54665 (54665), Seq: 7064875, Ack: 828, Len: 1460
> [3 Reassembled TCP Segments (2935 bytes): #19199(15), #19201(1460), #19202(1460)]
◢ Real Time Streaming Protocol
  ◢ Response: RTSP/1.0 200 OK$\000\005\260\200`\023\2222\336\305\035v:\316\222\\001\333\357\253\374\365\333\377\206\372\325\037\367\315\336;\227\333|^\\367\256Dsw\356\020\353(\216M/\311~nM\r
      Status: 200
    Data (2865 bytes)
```

Here the traffic is normal

8]Results:

1) Successful implementation of designed artificial neural net.
2) Verification of it in Genetec and Wireshark logs.
3) Enhanced features as seen below.



Video quality:

Resolution: H.264 1280 x 960

Quality: Custom

Constant bit rate: OFF

Image quality: 50 0 100 %

Frame rate: 30 1 45 fps

Key frame interval: 100 1 100 images

Recording frame rate: All frames

Crop image: OFF

Listing of noted features

a) **Resolution:h.264<full resolution obtained>**
b) **FPS:45(Optimal is 44)**
c) **Qos:4.86**

Appx:5

9]Future scope:a) Implementation in cloud security
b) Deep OGBF Routing tree penetration
c) This model could help us in developing the H.265 model in general.

References

[1] Website http://www.hindawi.com/journals/cin/2011/289398/fig2/
[2] Website: http://www.tcpipguide.com/free/t_TCPIPArchitectureandtheTCPIPModel-2.htm
[3] M. E. Smid and D. K. Branstad, “The Data Encryption Standard: Past and Future,” Proceedings of The IEEE, vol. 76, no. 5, pp. 550-559, 1988.
[4]An Introduction to Neural network" by Ben Krose and Patrick van der Smat Eighth edition November 1996