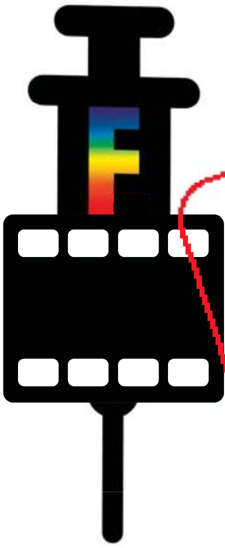
ראה סיכום הערות בסוף הדוח

אוניברסיטת ב
sity of the Negev



נבדק ללא ציון
נא לתקן הערות במסגרת הגשת דוח התקדמות

Pre-Report

CYBER-ATTACK BY PLAYING VIDEO ON IOS DEVICE

חסר כותרת משנה המרחיבה במקצת על הפרוייקט
חסר מספר הפרוייקט
חסר מספרי תעודות זהות ותפקיד שלכם
מיקום תאריך מתחת לכותרת

xxxxxxxxxxxxx xxxxxxxxxxxx

Instructors:

xxxxxxxxxxxxx xxxxxxxxxxxx

7.12.2018

Table of Contents

1 Abstract

הקדמה יפה ולעניין

The project is part of a comprehensive study by the research group headed by Prof. Hadar, which examines the suspicion of a security breach in various operating systems and offers protection against it. The security breach is the assimilation of a hidden channel within a compressed video stream that enables sending commands remotely to a device that plays the video.

So far, the security vulnerability in Linux and Android operating systems has been studied, and as part of this project we will focus on the response of the iOS operating system to this type of attack.

We'll demonstrate ways to take over iPhones with video playback in a dedicated app written to iOS. The player will decrypt the malicious code that was planted in a hidden channel and execute commands that will cause the device to be subject to a remote control.

The video will be played as a stream from an external server that will plant the video malware. The app will use a filter to read the code and make it executable.

נא להוסיף:
מכיוון שצוות המחקר מתמחה בדחיסת וידאו הרי ערוץ סומי הוא סוג של אימלמנטציה של דחיסה. שכן ערוץ סמויי זו צורה אחרת של דחיסת וידאו.. שכן אם ניתן לשתול מידע מבלי שנבחין הרי שלחלופין אפשר לדחוס יותר טוב ואז לא יהיה ניתן לשתול את המידע

1.1 Key words:

Multimedia compression, video, encoding, iOS, filter, FFmpeg, malicious code, hidden channel, operation system, header, payload, cyber, DCT

2 Introduction and motivation

Nowadays, we are witnessing the increasing consumption of multimedia content in general, and specifically, of video services worldwide. With the increment of accessible and high-quality online video services, we need to comply with this by compression standards that significantly reduce storage size and bandwidth usage like H.264, JPEG, H.265 etc.

The endless pursuit between the attacker and the defender only sharpens the levels of sophistication, aggressiveness and resources of attacks. In view of all the above, platforms for the transmission of large digital information over the Internet are very useful for hackers because it's very difficult to detect malicious code hidden inside a video stream.

יש להרחיב ולהסביר מה המטרה של התוקף ומה המטרה של המגן.
יש להביר שמשחק התפקידים בין התוקף למגן מתחלף ותולי אפליקציה.
למשל בשתילת תוכנה זדונית התוקף מנסה לשבש את פעול המכשיר הסדירה.
בזכויות יוצרים המגן דווקא שותל מידע חיובי ואילו התוקף מנסה לשבש את החתימת מים..

3 The problem

Most of the methods of cyber-attack by video are using the header of the packets transmitted over the network. This method is known and more easily identifiable, therefore unlikely to succeed. In addition, the amount of information that can be transmitted in such an attack is limited.

Nowadays anti-virus software can scan the packet's header and identify patterns of malicious code, so we need to find a way to implant the code inside the video data. Hiding the malicious code in a RAW file is not possible because after compressing the file, the compression algorithm commonly DCT causes the bits to be reverted in the RAW file. Therefore, an attacker cannot use the malicious code by this method.

In addition, since DCT-based watermarking attacks are known as described in [1] and [2].

A different method of hiding malicious information in Multimedia is needed.
Emphasis should be placed on non-message information such as watermark as shown in Figure 1.
Therefore, we choose to send information that is known opcodes to the attacker and the software of the attacked party only. Therefore, it would be more difficult to understand such an attack.

למרכז שרטוטים

Figure 1- Example of watermark method [2]

However, the correct use of the compression algorithm makes it possible to embed malicious information as an integral part of the video's payload. This method, if performed wisely under restrictive conditions, is undetectable.

There are several challenges for an attack algorithm to deal with:

- Real time processing: Cause minimal delay and take into consideration video rate and quality.
- Perceptual quality: Maximizing the amount of embedded data while minimizing the video quality degradation. Keep user quality of experience high.
- Keep video size and bitrate closest to the uncontaminated video: The disruption of malicious data can increase the stream bit rate or size, which can alert a defense system.
- Be able to execute the malicious code without arouse any suspicion by the operating system (iOS).
- Power efficiency: Because we are implementing the algorithm on a mobile device without an infinite power source, we want to decrease the power consumption as much as possible.

This attack of inserting malicious data in the payload is harder to detect because the infected multimedia content is well formed, and the quality of the infected content is not degraded, in oppose to an attack that use the video header to conceal data. Using the first method and concealing the data in the compressed domain enables the attack to occur in real time, under certain restrictions.

4 Project Goal

The main focus of this project is to supply a proof of concept to a cyber-attack that conceals malicious data in a video payload on iOS device, in the compressed domain, by means of steganography (in real time) on one hand, and extracting this malicious data by using a covert channel and a malware (that had been previously planted in an end user side), on the other.

The suggested attack will be implemented in the H.264 standard since it is widely used and offer flexibility in the compression process. H.264 standard defines syntax for compressed video and a method for decoding this syntax to produce a displayable video sequence.

The covert channel that connects the malware and the adversary will be the locations of DCT coefficients, a known dictionary and the malicious data will be concealed within the coefficients value.

In order to measure the quality of the attack I will perform several tests to ensure high accuracy in detection of the malicious data upon receiving the infected H.264 bitstream. Other quality metrics used are the well-known MSE (mean square error) and PSNR (peak signal to noise ratio) metrics to ensure that the additive infected data don’t increase bitrate to a noticeable level.

אין סיבה להשאיר ריק.
אפשר להתחיל את המקטע
הבא

5 Tools and Software

5.1 FFmpeg 4.1

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines h[illegible]eate[illegible]upports the most obscure ancient formats up to the cutting edge. No matt[illegible]were designed by some standards committee, the community or a corp[illegible]It is also highly portable: FFmpeg compiles, runs, and passes our testing inf[illegible]ct[illegible]ATE across Linux, Mac OS X, iOS, Microsoft Windows, the BSDs, Sol[illegible]is, etc. u[illegible]der a wide variety of build environments, machine architectures, and configurations.

FFmpeg is part of the workflow of hundreds of other software projects, and its libraries are a core part of software media players such as VLC, and has been included in core processing for YouTube and the iTunes inventory of files. Codecs for the encoding and/or decoding of most of all known audio and video file formats is included, making it highly useful for the transcoding of common and uncommon media files into a single common format.

5.2 Xcode 10.1

Xcode is an integrated development environment (IDE) for macOS containing a suite of softw[illegible]re development tools developed by Apple for developing software for [illegible]acO[illegible]S, watchOS, and tvOS. First released in 2003, the latest stable release is [illegible]0.1 and is available via the Mac App Store for macOS High Sierra and [illegible]Mojave users. Using the iOS SDK, Xcode can also be used to compile and [illegible]ebug[illegible]ications for iOS that run on ARM architecture processors.

We'll use Xcode to develop a video player application that runs on the victim's device, which will execute the malicious code sent by the attacker.

5.3 FFmpeg iOS build script

This is a shell script to build FFmpeg libraries for iOS and tvOS apps. We need the FFmpeg liberty in our iOS app for manipulate the income video stream and extract the [illegible]licio[illegible]de from it. Because there is no any built-in support for FFmpeg, we need [illegible]pile the FFmpeg source code by using this script.

5.4 VMware Workstation 15

VMware Workstation is the industry standard for running multiple operating systems as virtual machines (VMs) on a single Linux or Windows PC. IT professionals, developers and businesses who build, test or demo software for any device, platform or cl[illegible]ely on Workstation Pro.

[illegible]use Xcode works only with mac OS X computers, we'll use VMware for running [illegible] on mac OS X sierra VMware image.

5.5 Visual Studio 2017

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web[illegible]vices and mobile apps. Visual Studio uses Microsoft software development [illegible]ns such as Windows API, Windows Forms, Windows Presentation [illegible]dation, Windows Store and Microsoft Silverlight. It can produce both native [illegible]d managed code.

We used Visual Studio for developing program called YUVeditor to manipulate RAW video files in YUV 4:2:0 format.

5.6 HxD 2.1

HxD is a freeware and lightweight file Hex editor for Windows. We used this s[illegible]re to understand file structures on the non-compressed plane. We also used it to [illegible] what changes were made after standard H264 compression, and how many bits [illegible]e changed.

6 Theory Background

6.1 YUV format

YUV is a method for encoding color image and video This coding is designed with the knowledge about the human eye and on its disability to discern the smallest details. One of the main purposes of this technique is to reduce bandwidth for chrominance details. In addition, this technique was developed to better handle data errors resulting from random compression and error processes compare to the RGB representation.

The Y'UV color space format is defined by three component's: one for the luminance (Y) and two for the chrominance (U and V) (see Figure 2).

In the past, only the Y component was used, meaning black and white only.
Then, the U and V component was added separately from the color information with a sub-carrier for the backward compatibility (for the black and white devices).

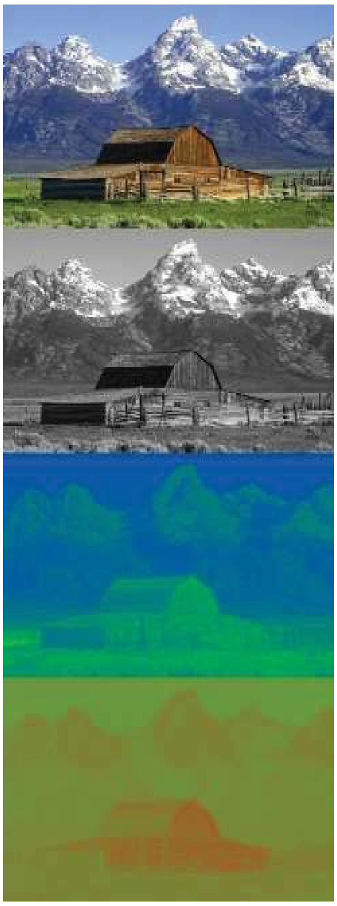


Figure 2 - An image along with its Y', U, and V components respectively (Wikipedia)

7 Method description

7.1 Planting malicious data in the RAW plane

We will first examine the feasibility of an attack using malicious code that will be transferred as hidden commands within the video file by an agreed protocol between the attacker and the affected device.

Since the compression of information from the source will not completely restored, we will aggressively pass commands and control commands to the victim device so that the commands can be decoded even after a partial loss of information. On the receiving side, the app will regularly check the agreed area of the planted commands.

Any commands we want to send will be represented by pixels in the video. Consider the color of the pixels and order their appearance to see which command was sent. After identifying the command, the victim's device will execute the command accordingly.

In this way, we can send and even receive information back in the same way without arousing suspicion of traffic between the victim's device and the attacker's server.

7.2 Graphic description of the attack

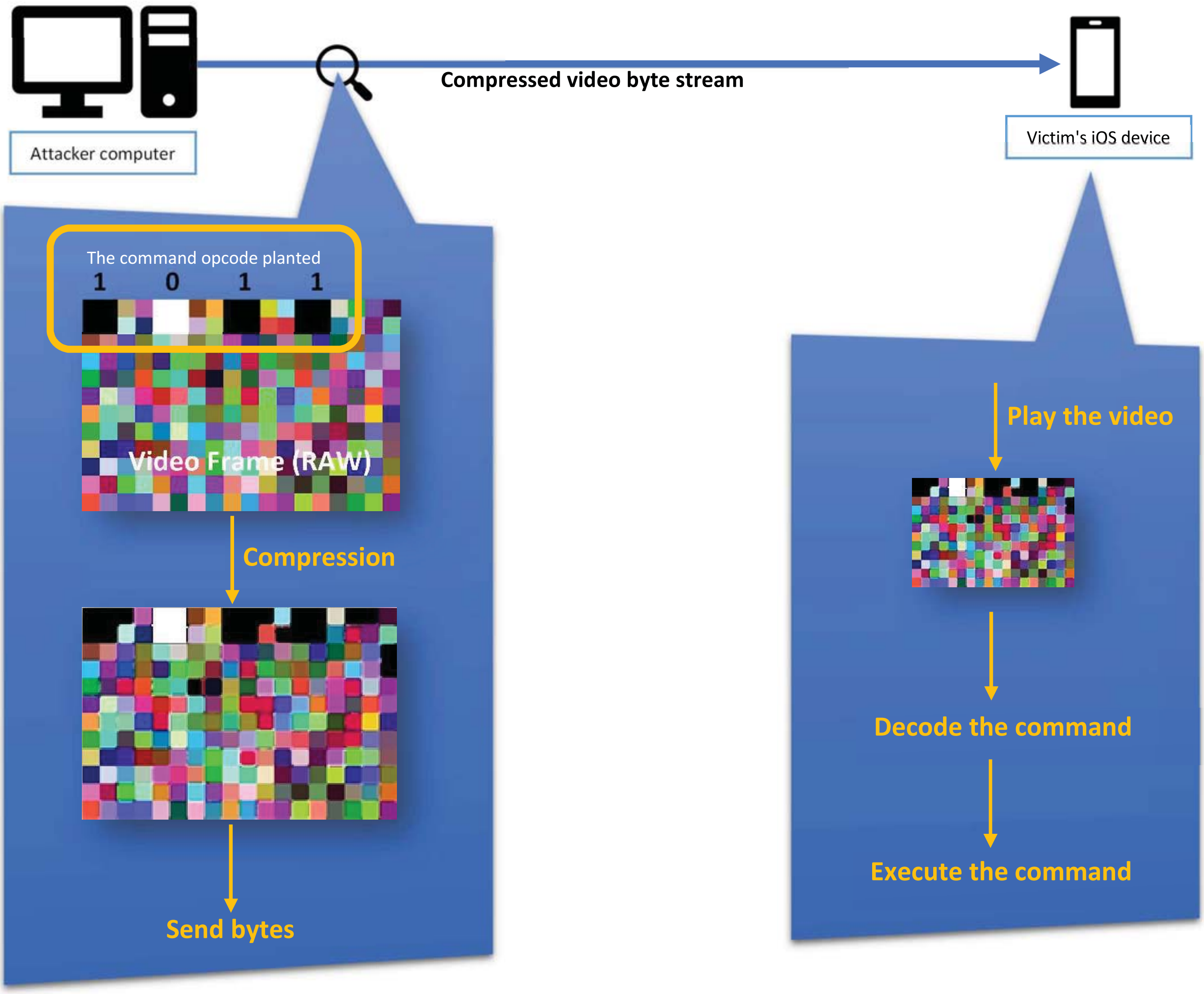


Figure 3 - The attack idea graphically explained

שרטוט יפה מאד. מיספור השרטוט צריך להיות במרכז

7.3 Get pixel position

First, we'll show how to calculate the position of a video's YUV components position in the byte-stream. Suppose a video file with known resolution (width x height) in pixels.

The data resolution of a RAW YUV file is 2x2 pixels square, so the total amount of these squares is $\mathrm{C} = \frac{\mathrm{width} \times \mathrm{height}}{4}$. In YUV 4:2:0, the square consists of 6 bytes: 4 bytes for the Y, one for U and one for V. Now it is clear that the whole frame size is $\mathrm{C} \times 6$ bytes, where the total size of the Y values is $\mathrm{Y} = \mathrm{numOfcolors} \times 4$.

To reach the (x, y)[1] coordinate in the video:

$\mathrm{Y}_{\mathrm{x,y}}$ **first** pair position in file $= \mathrm{Frame}[\mathrm{x} \times \mathrm{width} + \mathrm{y}]$

$\mathrm{Y}_{\mathrm{x,y}}$ **second** pair position in file $= \mathrm{Frame}[\mathrm{x} \times \mathrm{width} + \mathrm{y} + \mathrm{width}]$

$\mathrm{U}_{\mathrm{x,y}}$ position in file $= \mathrm{Frame}[\, Y + \frac{\mathrm{x}}{2} \times \frac{\mathrm{width}}{2} + \frac{\mathrm{y}}{2}]$

$\mathrm{V}_{\mathrm{x,y}}$ position in file $= \mathrm{Frame}[Y + \frac{\mathrm{x}}{2} \times \frac{\mathrm{width}}{2} + \frac{\mathrm{y}}{2} + \mathrm{C}\,]$

השימוש בנוסחאות הוא מעולה.
רק חייב להכתב בצורת הוכחה. כלומר לכל נוסחה נותנים מספר והמלל הוא מתחת או לפני. לא לשלב אחד בתוך השני. במלל יש להזכיר את מספר הביטוי המתמטי.

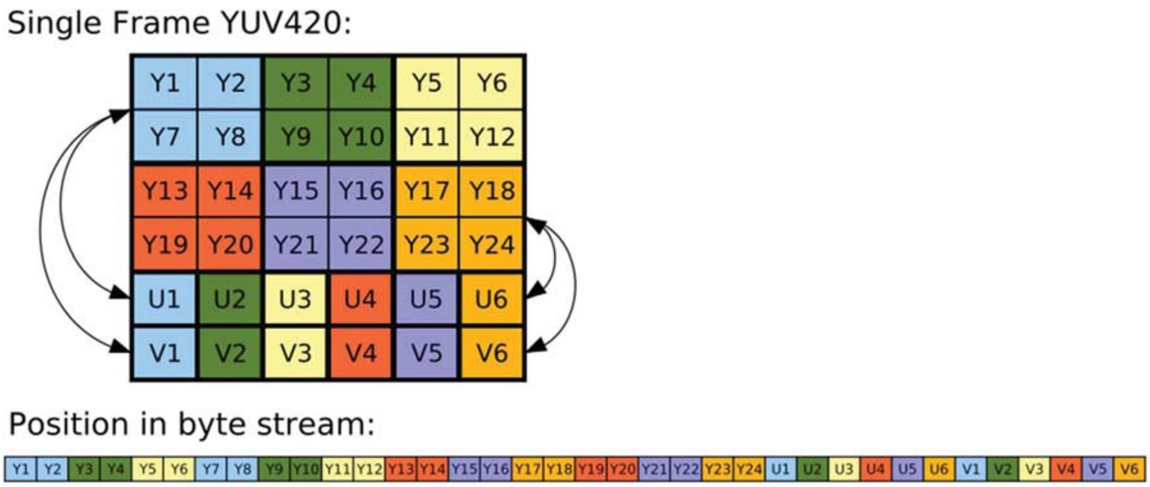


Figure 4 - The Y', U and V components arrangement in Y'UV420 file format (Wikipedia)

[1] Where $x = 2k\,, 0 \le x \le width\,, k \in \mathbb{N}$ and $y = 2n\,, 0 \le y \le height\,, n \in \mathbb{N}$

7.4 Transplant opcode in pixels

Now that we know how to access each pixel, we will create a uniform and agreed pattern of command code that we would like to transmit with the video. For example, we will define a 3-bit command space, and each command will be translated into actual action on the victim's device.

Table 1 - Opcode as commands

Opcode	Task
001	Take picture
010	Turn on GPS
011	Go to URL

We will embed the commands in pixels by a FFmpeg filter we will write. One filter will run on the attack server, which encode the command, and the other decode filter will be in the victim iOS device, executing the commands.

8 Completed Tasks

8.1 Learn to use FFmpeg via CLI

The FFmpeg software does not include a graphical user interface, but only a command line interface, by which you can play video files, convert, compress, use filters and more. Now, we will demonstrate a few basic commands:

1. Playing a video file, for example we want to play the file 'vid.mp4' (H.264 format):

```
ffplay vid.mp4
```

2. Show all the available pixel formats:

```
ffmpeg -pix_fmts
```

3. Convert the mp4 video file to RAW data:

```
ffmpeg -i vid5.mp4 -c:v rawvideo -pix_fmt yuv420p
bird.yuv
```

*We used the "YUV 4:2:0" pixel format

4. Compress the RAW video file to mp4 (H.264):

```
ffmpeg -f rawvideo -pix_fmt yuv420p -s:v 1280x720 -r 25 –
i bird.yuv -c:v libx264 output.mp4
```

*Notice that we must put manually the original video resolution because this info can't be found in the RAW file.

5. Use a filter that flips the video vertically:

```
ffmpeg -i vid5.mp4 -vf "split [main][tmp]; [tmp]
crop=iw:ih/2:0:0, vflip [flip]; [main][flip]
overlay=0:H/2" output.mp4
```

We copied the main flow, cut it in half, we turned it vertically ("vflip") and showed it as overlay on the main flow.

6. Play a video while applying a filter in real time:

```
.\ffplay.exe -i .\original.mp4 -vf
"drawbox=0:3:10:70:green@0.4:t=fill"
```

This filter can draw boxes in different colors over the video frames.

8.2 Manipulating RAW video files

After de-compressing a video by using FFmpeg, we want to edit the file, so that after compression and viewing, there will be disruptions in the frame.

First, we used HxD in order to disrupt frames in the video. We inserted random values in the YUV 4:2:0 RAW file, inside one frame, then we compressed it and the result can be shown in Figure 5.

Figure 5 - frame is corrupted by inserting random values

The previous example was intuitive, and now we want to edit each pixel as we want, in each color and frame. To this end, we have developed a C program called "YUVeditor" which accepts as input a compressed video file in YUV 4:2:0 format, the resolution of the video and the name of the output file to be created.

To use the program, we write this in the command line. The first argument is the original RAW video file, the second is the output RAW file, the third is the video width and the last is its height.

```
Yuveditor.exe originalvid.yuv outvid.yuv 1280 720
```

The program will automatically draw rectangle over the frames, compress the video with FFmpeg and play the video to the user with FFplay. You can see the operation in Figure 6 - YUVeditor sequenceFigure 6.

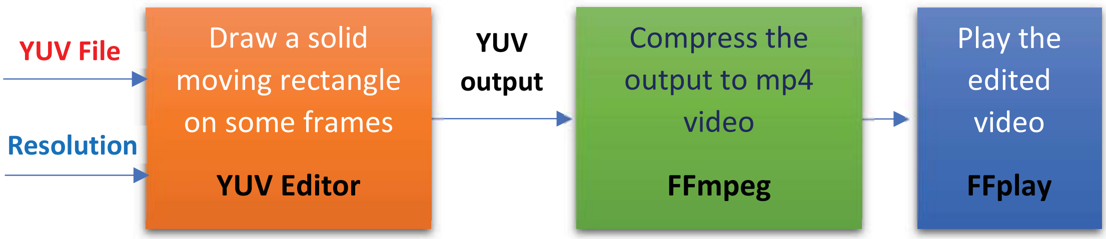


Figure 6 - YUVeditor sequence

מעולה

For example, we ran the program with a short sample video that can be seen in Figure 7. You can see the result in Figure 8. The green rectangle is created as a result of the change in values made by the software.

Figure 7 - Original video as input to YUVeditor

Figure 8- Edited video after YUVeditor proccess

Here is the code of the function that changes the YUV values in the RAW data:

```
void extractYUV(YUVFile * yuvfile, int x, int y, YUV * color, int frameNumber) {

    int u, v;
    int numOfColorsInline, sizeOfDoubleLine;
    int yFirstPair, ySecondPair;

    yFirstPair = (x)*yuvfile->width + y;
    ySecondPair = yFirstPair + yuvfile->width;
    u = yuvfile->sizeOfY + (x / 2) * (yuvfile->width / 2) + y / 2;
    v = u + yuvfile->numOfcolors;

    //y first line
    yuvfile->filedata[yuvfile->framesize*frameNumber + yFirstPair] = color->y;
    yuvfile->filedata[yuvfile->framesize*frameNumber + yFirstPair + 1] = color->y;
    //y second line
    yuvfile->filedata[yuvfile->framesize*frameNumber + ySecondPair] = color->y;
    yuvfile->filedata[yuvfile->framesize*frameNumber + ySecondPair + 1] = color->y;
    //u
    yuvfile->filedata[yuvfile->framesize*frameNumber + u] = color->u;
    //v
    yuvfile->filedata[yuvfile->framesize*frameNumber + v] = color->v;
}
```

We have calculated the YUV position values according to the calculation shown in *Get pixel position* section.

8.3 Writing a filter and compile it for FFmpeg

We need to write a filter ourselves to implement the attack method for each video we want to stream to the user. The filter will color pixels as opcode for the command we want to transmit and execute. We managed to compile the whole FFmpeg code into an executable program with our own filter.

פה דרושה הרחבה. מה זה
פילטר, איך הוא משתלב,
מה האינטרפייס אליו וכיצד
תעבדו איתו

9 Schedule and Phases

Table 2- Project schedule

טבלת משימות מעולה

Task	Final Date	Days
Define the project	20/10/2018	Done
Literature survey - understanding the attack method	21/10/2018	Done
Submit proposal for the project with an abstract	25/10/2018	Done
Learn how to use FFmpeg and manipulate video files	30/10/2018	Done
Write our own FFmpeg filter	05/11/2018	Done
Submit pre-Report	30/11/2018	Done
Define coding by visiable pixels protocol	01/12/2018	1.00
Hide malicious code in video by FFmpeg filter using the above protocol	10/12/2018	10.00
Learn how to code in iOS (Swift)	19/12/2018	19.00
Compile FFmpeg for iOS	28/12/2018	28.00
Develop our own application for iOS	06/01/2019	37.00
Submit progression report	11/01/2019	42.00
Implement the attack in iOS device	24/01/2019	55.00
Check possible visible attacks on iOS devices	02/02/2019	64.00
Check the application with streamed video from our server	11/02/2019	73.00
Hide malicious code in video in the DCT plane	20/02/2019	82.00
Sign up for "Engineering project B"	24/02/2019	86.00
Implement the attack in iOS device	01/03/2019	91.00
Check possible unvisible attacks on iOS devices	19/03/2019	109.00
Submit presentation	19/03/2019	109.00
Check the application with streamed video from our server	28/03/2019	118.00
Update project name	06/04/2019	127.00
Submit summary	15/04/2019	136.00
Submit poster	24/04/2019	145.00
Filling file requirements for equipment to conference	03/05/2019	154.00

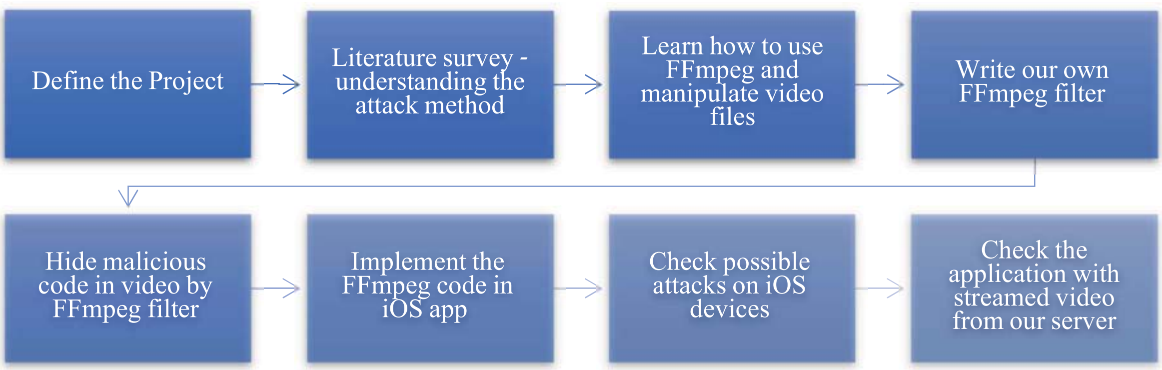


Figure 9 – Milestone

אבני דרך יש להכין בצורת טבלה.
כל קבוצת משימות מובילה להשלמת אבן דרך. לכל
אבן דרך יש לציין תאריך סיום ותוצר שניתן למדידה.

פרק זה יש להוסיף כנספח לפרוייקט

10 Technical information

10.1 Google Drive Link

https://drive.google.com/drive/u/1/folders/1CTEgcNufNzvxeXre-8ptfet3p7qGky1I

10.2 Hardware

Apple iPad pro 12.9"

Table 3 - iPad pro 2018 specifications

Display **12.9"** Liquid Retina display 12.9-inch (diagonal) LED-backlit Multi-Touch display with IPS technology 2732-by-2048-pixel resolution at 264 pixels per inch (ppi) ProMotion technology Wide color display (P3) True Tone display Fingerprint-resistant oleophobic coating Fully laminated display Antireflective coating 1.8% reflectivity 600 nits brightness **Chip** A12X Bionic chip with 64-bit architecture Neural Engine Embedded M12 coprocessor **Camera** 12-megapixel camera ƒ/1.8 aperture Digital zoom up to 5x Five-element lens Quad-LED True Tone flash Panorama (up to 63 megapixels) Sapphire crystal lens cover Backside illumination sensor Hybrid IR filter Autofocus with Focus Pixels Tap to focus with Focus Pixels Live Photos with stabilization Wide color capture for photos and Live Photos Improved local tone mapping Exposure control Noise reduction Smart HDR for photos Auto image stabilization Burst mode Timer mode Photo geotagging Image formats captured: HEIF and JPEG **Video Recording** 4K video recording at 30 fps or 60 fps 1080p HD video recording at 30 fps or 60 fps 720p HD video recording at 30 fps Quad-LED True Tone flash Slo-mo video support for 1080p at 120 fps and 720p at 240 fps Time-lapse video with stabilization Cinematic video stabilization (1080p and 720p) Continuous autofocus video Noise reduction Take 8-megapixel still photos while recording 4K video Playback zoom Video geotagging Video formats captured: HEVC and H.264	**TrueDepth Camera** 7-megapixel photos Portrait mode Portrait Lighting Animoji and Memoji 1080p HD video recording at 30 fps or 60 fps Retina Flash ƒ/2.2 aperture Wide color capture for photos and Live Photos Smart HDR Backside illumination sensor Auto image stabilization Burst mode Exposure control Timer mode **Video Calling**[3] FaceTime video iPad to any FaceTime-enabled device over Wi-Fi or cellular **Audio Calling**[3] FaceTime audio iPad to any FaceTime-enabled device over Wi-Fi or cellular **Speakers** Four speaker audio **Microphones** Five microphones for calls, video recording, and audio recording **Cellular and Wireless** **All models** Wi-Fi (802.11a/b/g/n/ac); simultaneous dual band (2.4GHz and 5GHz); HT80 with MIMO Bluetooth 5.0 technology **Location** **All models** Digital compass Wi-Fi iBeacon microlocation **Wi-Fi + Cellular models** Assisted GPS, GLONASS, Galileo, and QZSS Cellular **Sensors** Face ID Three-axis gyro Accelerometer Barometer Ambient light sensor

10.3 Operation systems

1. Microsoft Windows 10
2. Apple mac OS X high sierra
3. Ubuntu 17.04

11 References

[1] W. Peipei, C. Yun, Z. Xianfeng and Z. Meineng, "A Steganalytic Algorithm to Detect DCT-based Data Hiding Methods for H.264/AVC Videos," *IH&MMSec,* 2017.

[2] L. Xin, W. C. Xingjun and X. Linghao, "A Simplified and Robust DCT-based Watermarking Algorithm," *2nd International Conference on Multimedia and Image Processing,* 2017.

[3] Wikipedia, "YUV," [Online]. Available: https://en.wikipedia.org/wiki/YUV.

[4] FFmpeg, "FFmpeg," [Online]. Available: https://ffmpeg.org/ffmpeg-filters.html.

[5] M. Edry and L. Odiz, "A cyber attack using a covert video channel on the H.264 compressed," 2016.

[6] L. Yahav, "DCT-based Cyber Attack in the H.264 Coding Standard," 2017.

יש לציין שזה נעשה
במסגרת פרוייקט סיום
במחלקת... באוניברסיטה...

12 Table of Figures

טבלת שרטוטים צריכה להופיע אחרי תוכן ענינים ולאחר מכאן
טבלת רשימת טבלאות

לסכום
דוח מושקע וערוך ברמה טובה.
אך דוח זה היה מקבל ציון נמוך כי יש פה מידע שהעתקתם ללא ציון המקור. יש
להסביר במילים שלכם מה זה עושה וכיצד תשתמשו בו.
להשלים טבלת אבני דרך כולל תאריכים ותוצרים
חסר טבלת קיצורים ומושגים
נא לכתוב פיתוח מתמטי בצורה מקובלת
אין התייחסות לסיכונים, ומה תעשו אם הם יתרחשו
כדאי להוסיף תאור בדוח של מה יהיה התוצר הסופי של הפרוייקט
all pages should include headers and footer
נא לצרף שרטוט גאנט (שרטוט המתאר משימות ולוחות זמנים בצורה גראפית, חובה
לציין נקודות קריטיות- משימה שתלויה במשימה אחרת)