Seeing is Not Believing:

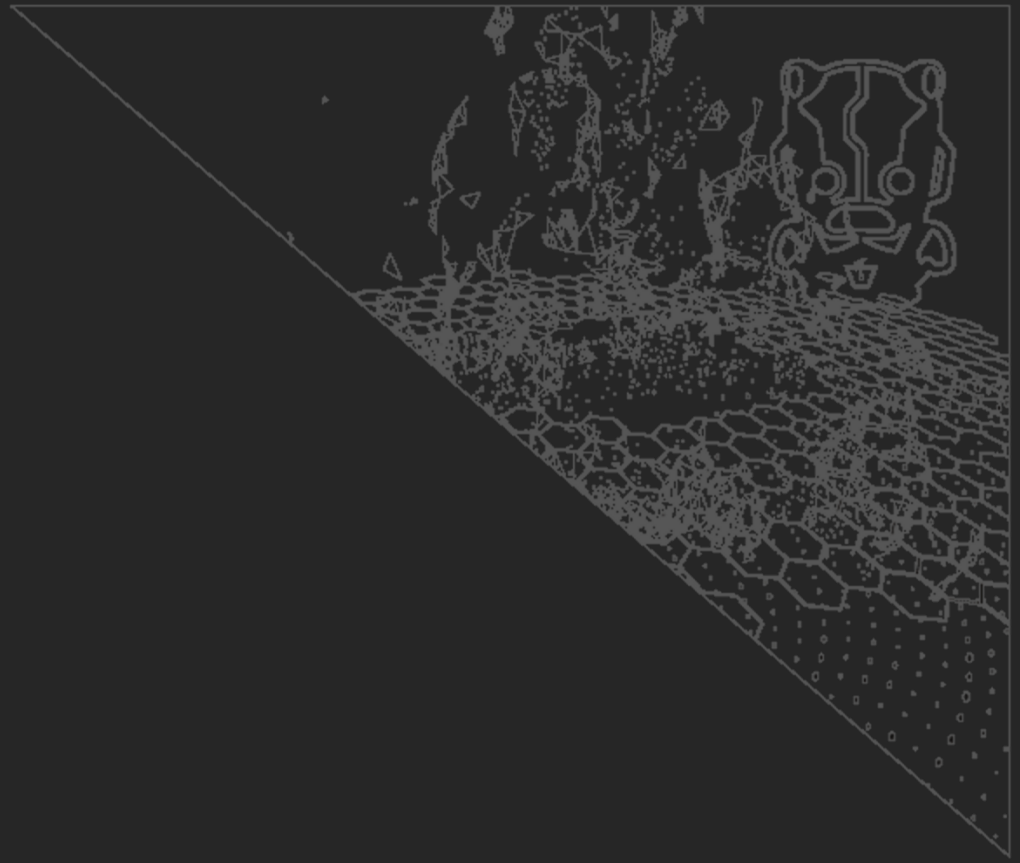
Bypassing Facial Liveness Detection by Fooling the Sensor

Elvin Gentiles

September 26, 2024

IOActive®

Introduction



whoami

- Senior Security Consultant @ IOActive
- Into identity verification and malware development
- CRTE, CRTO, CRTP, OSCE, OSCP, OSWP

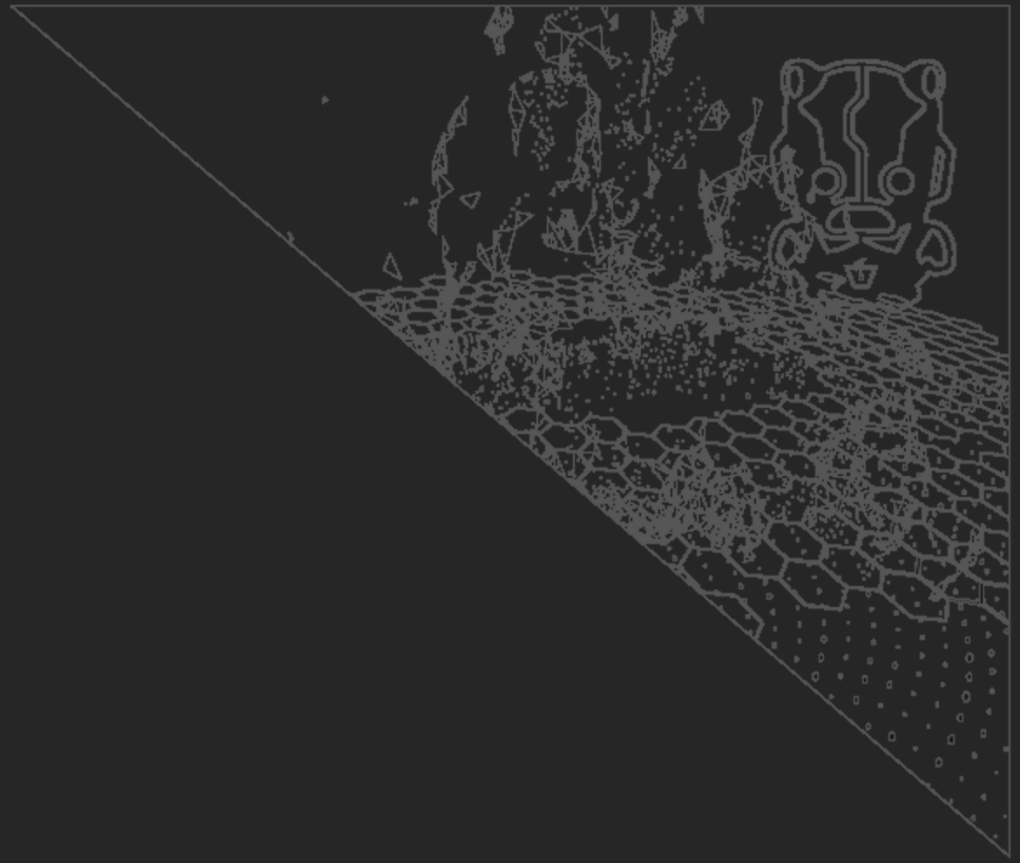
captmeelo.com
@CaptMeelo
capt-meelo
/in/elvingentiles

IOActive.

Agenda

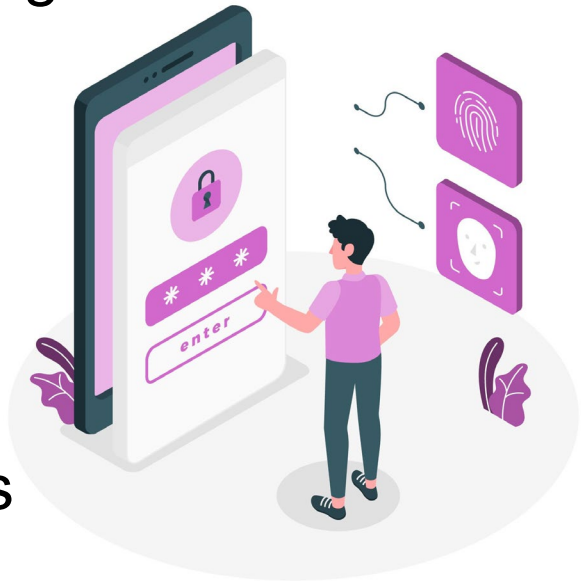
- Identity Verification
- Facial Recognition
- Facial Liveness
- Problems and Ideas
- The (Simple) Solution
- Threats
- The Way Forward

Identity Verification



Identity Verification

- A process to combat fraud by ensuring the end-users are real and who they claim to be.
- It plays a big role in Know Your Customer (KYC) and Anti-Money Laundering (AML).
- Common on highly sensitive applications and organizations, such as banking, crypto, etc.



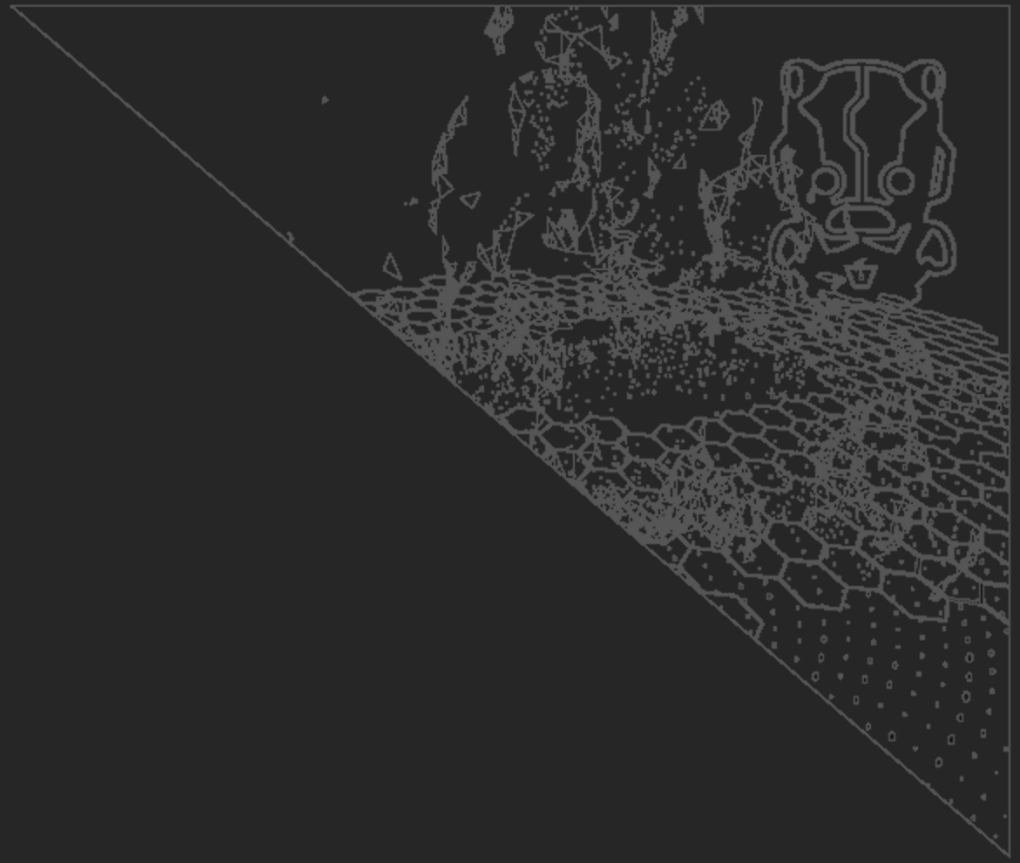
IOActive.

Types of Identity Verification

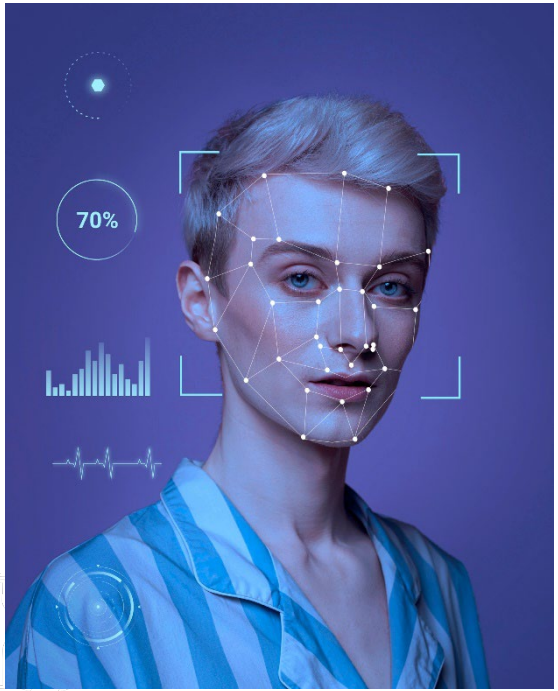
- Knowledge-based (password, pin, security questions)
- MFA (OTP)
- Document (Government ID, Driver's License, Passport)
- Biometric (Face, Fingerprint, Retina, Voice)

IOActive.

Facial Recognition



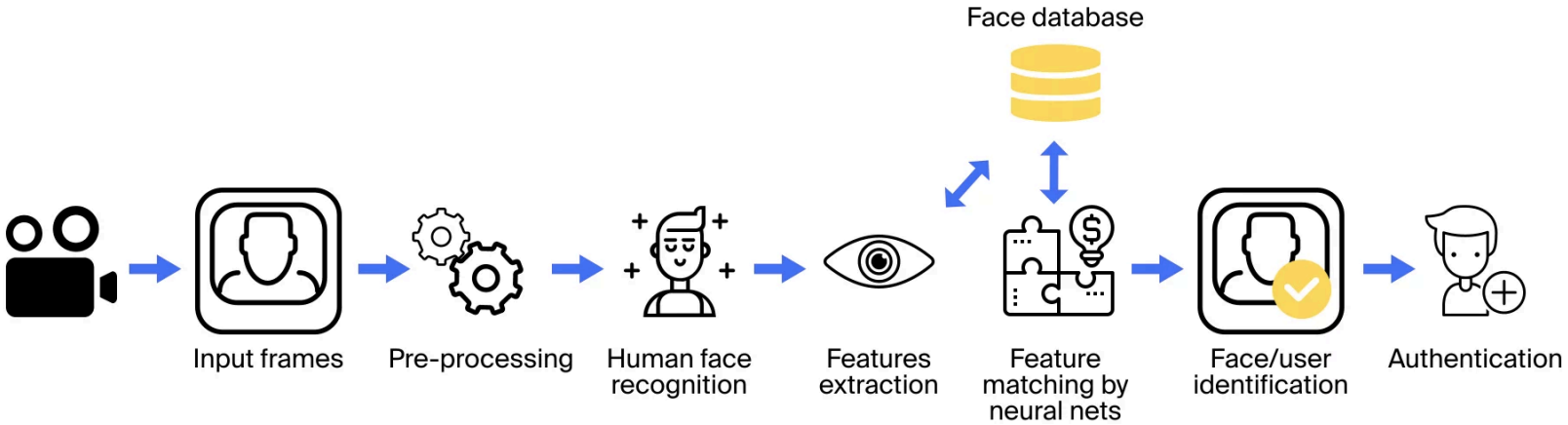
Facial Recognition



- A process whereby a user's facial features are captured, extracted, and compared against a database of faces to confirm identity.
- Answers the question, *"Is this the right person?"*

IOActive.

How Facial Recognition Works



Credits: https://hackernoon.com/things-you-need-to-know-before-installing-a-facial-recognition-system

IOActive.

Presentation Attack

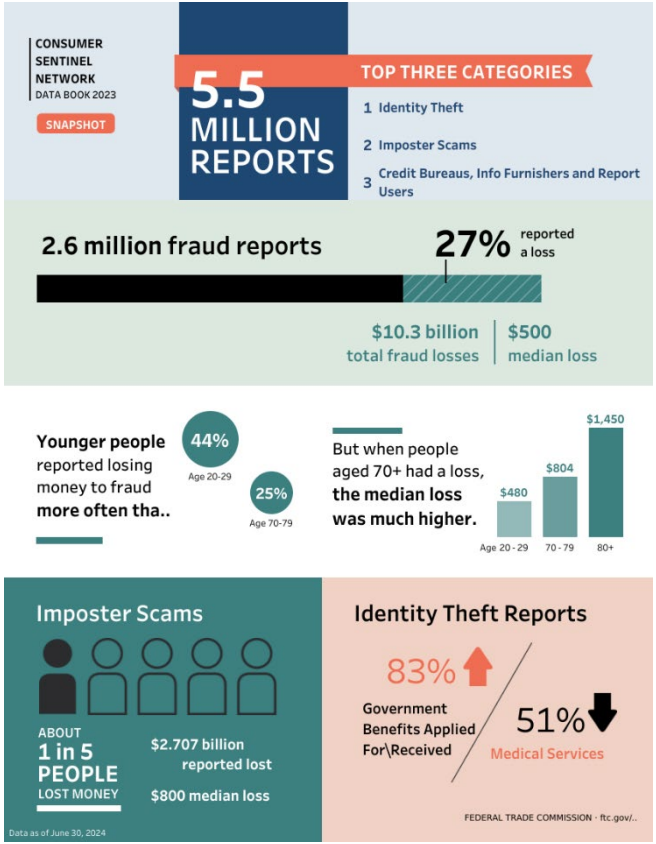
- a.k.a. “Facial Spoofing”.
- The attack is done by “spoofing” or “impersonating” a real person to fool the detection system.
- Typically carried out via:
  - **2D Spoofing**: Presenting a printed photo, or image displayed in a digital device (e.g., monitor or mobile phone) to the sensor/camera.
  - **Video Replay Attack**: Pre-recorded video is presented to the sensor/camera instead of a static image.
  - **3D/Silicon Masks**: A 3D reconstruction of a face is presented to the sensor/camera.

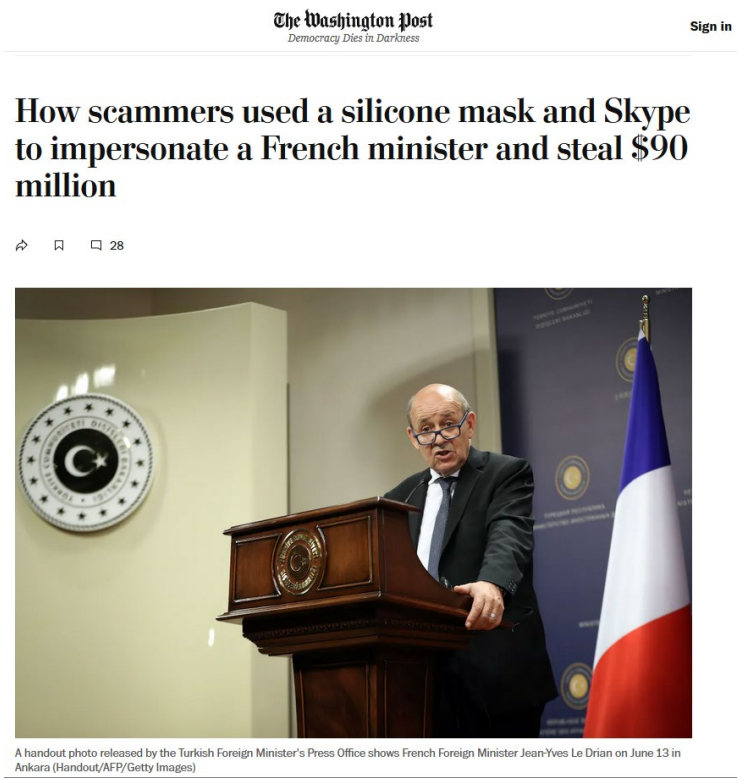
Presentation Attacks

Credits:
https://www.researchgate.net/publication/343150267_Learning_One_Class_Representations_for_Face_Presentation_Attack_Detection_using_Multi-channel_Convolutional_Neural_Networks

IOActive.

Real World Scenario





The Washington Post
Democracy Dies in Darkness

Sign in

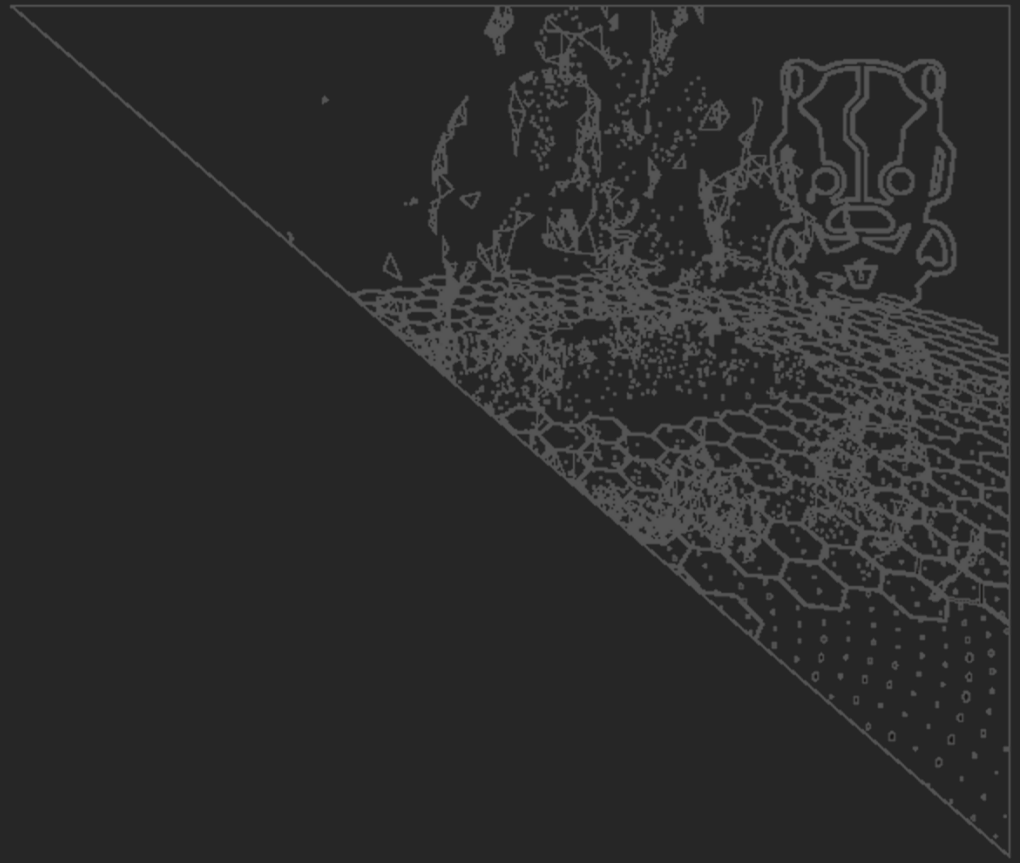
How scammers used a silicone mask and Skype to impersonate a French minister and steal $90 million

28

A handout photo released by the Turkish Foreign Minister's Press Office shows French Foreign Minister Jean-Yves Le Drian on June 13 in Ankara (Handout/AFP/Getty Images)

IOActive.

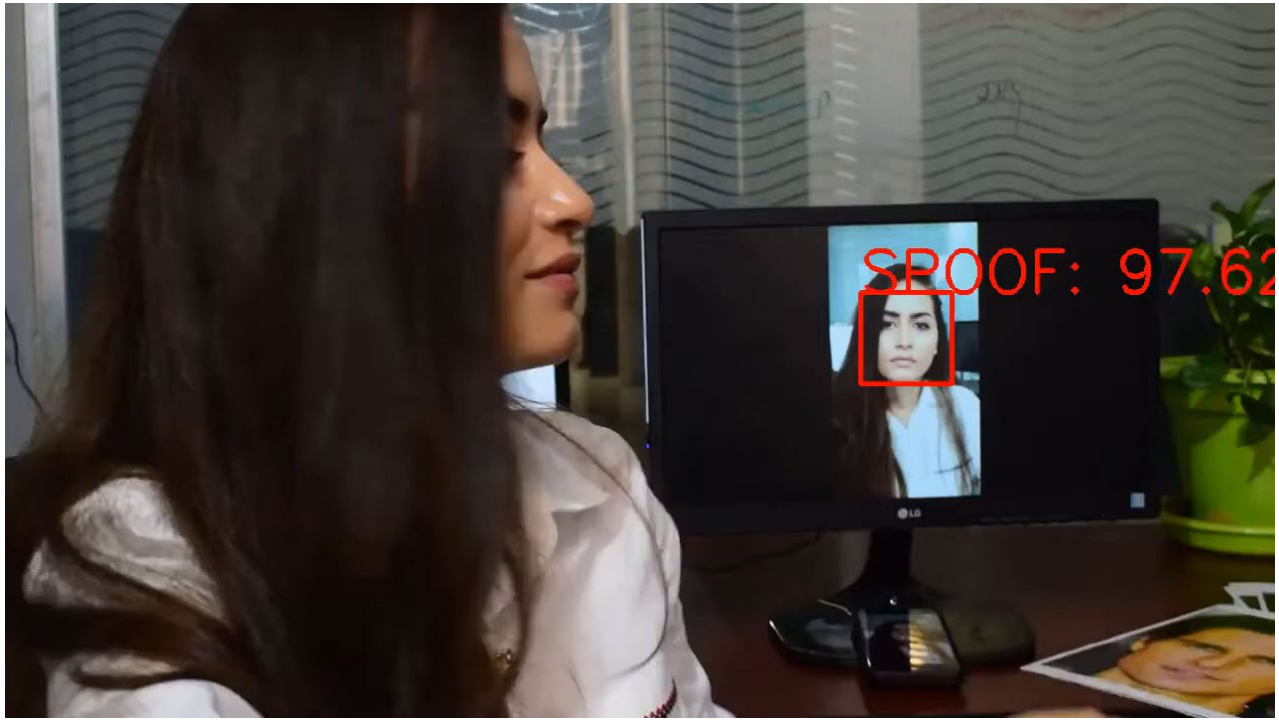
Facial Liveness



Facial Liveness

- A means to detect whether a biometric sample captured by a system belongs to a real, live human being or a fake representation.
- It works by capturing and analyzing a short selfie video or a stream of images to detect spoofs presented to the camera.
- Answers the question, “Is this a live person?”

Facial Liveness in Action



Credits: https://www.youtube.com/watch?v=FCuRob8stis (PresentID)

IOActive.

Passive vs Active Liveness

Passive:

- Does not require users' participation.
- Faster, efficient, and offers seamless user experience.
- Requires good input quality and might not be ideal in certain scenarios *(e.g., low light areas).*

Active:

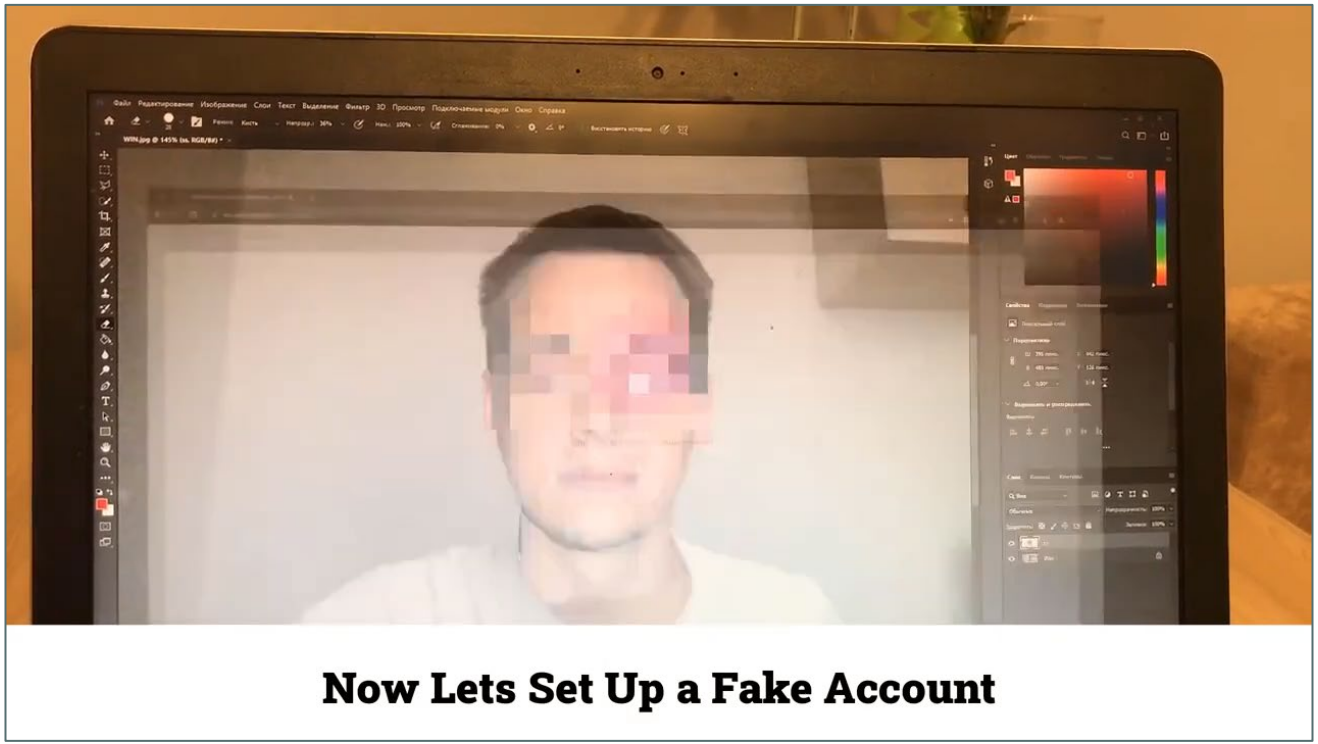
- Requires users to perform an action *(e.g., rotating the head, blinking, following a moving object).*
- Difficult to spoof with pre-recorded videos.
- Ensures the user is actively involved in the verification process.
- More tedious and inconvenient for users.

Passive vs Active Liveness in Action

Credits: https://www.youtube.com/watch?v=46_betoVH6Q (Sybrin)

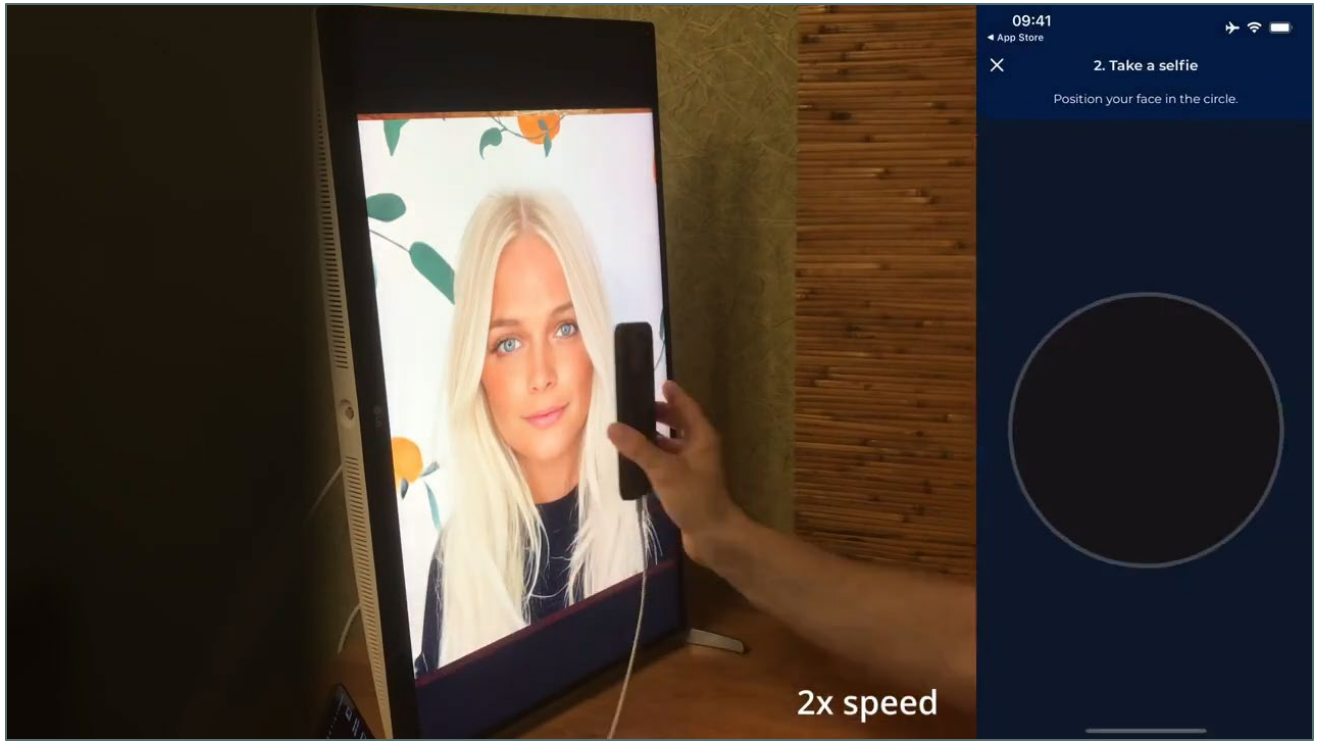
IOActive.

Successful Attacks Against Facial Liveness



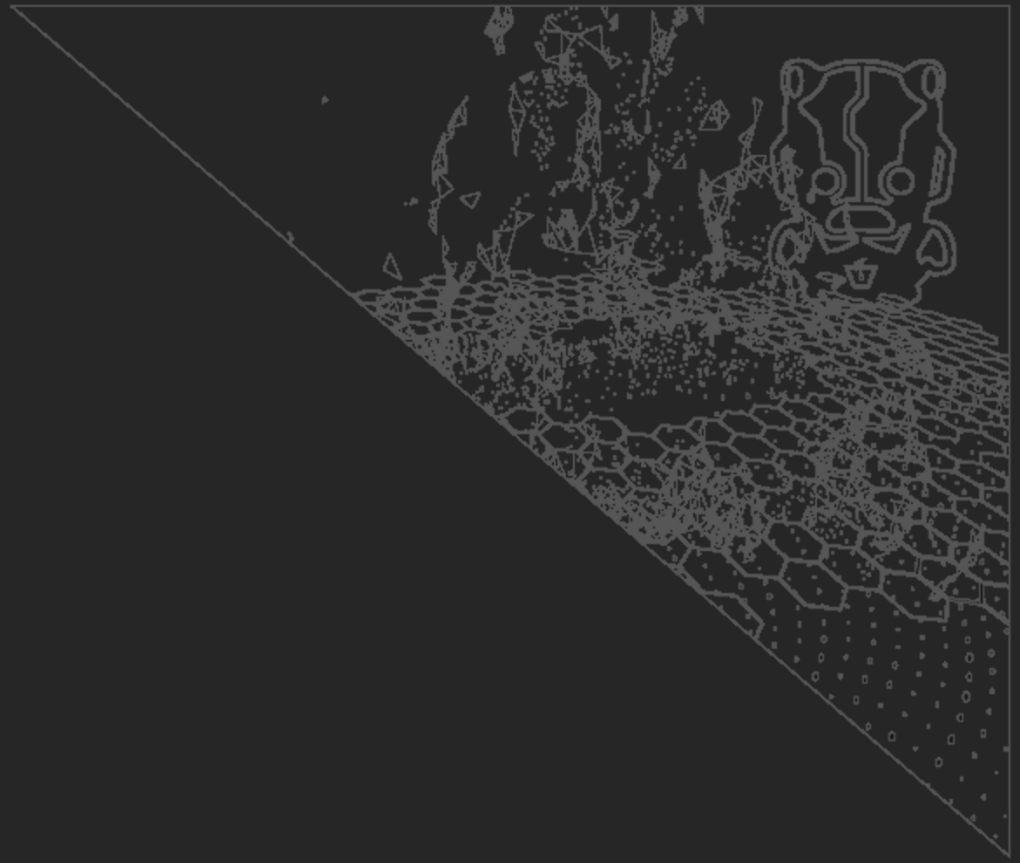
Credits: White Usanka

Successful Attacks Against Facial Liveness



Credits: White Usanka

Problems and Ideas



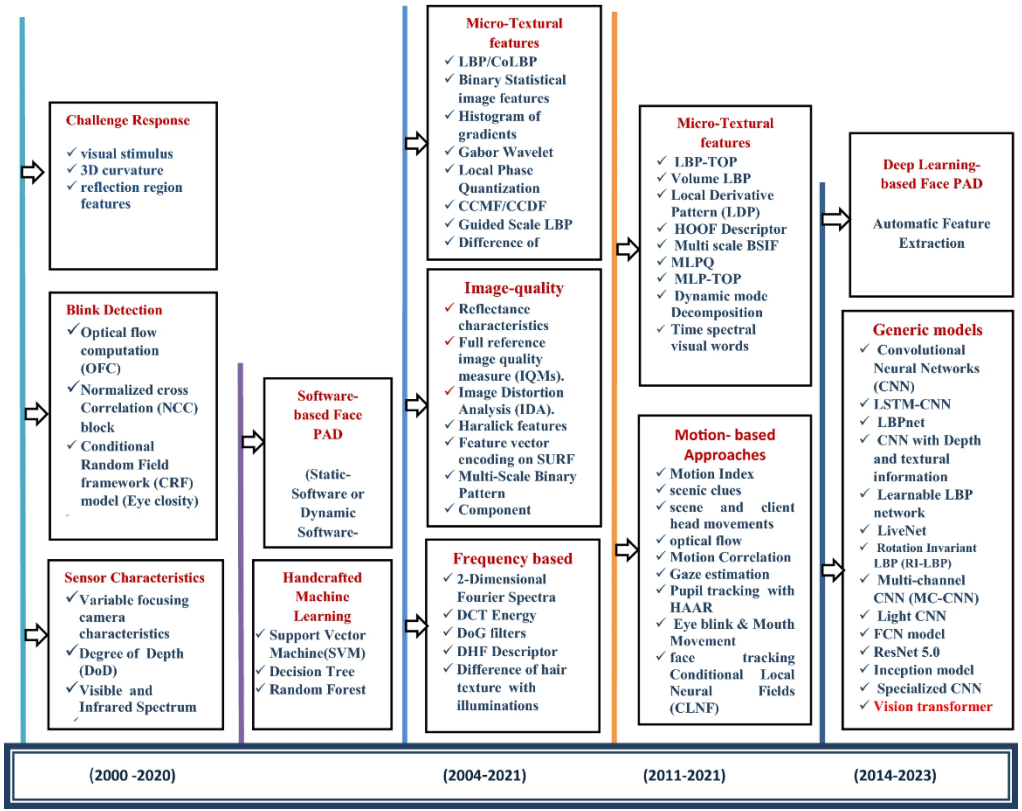
Continuing with the Previous Attack Scenario

- Using a display/monitor to present the image/video source to the sensor/camera (**video replay attack**)
- The method failed despite being only released last year (May 2023).
- Providers are stepping up their game.

Credits:
https://visagetechnologies.com/face-anti-spoofing-face-recognition/

IOActive.

Evolution of Presentation Attack Detection (PAD)



IOActive.

Recaptured Image Detection

Face liveness detection with recaptured feature extraction

Publisher: IEEE Cite This PDF

Xiao Luan ; Huaming Wang ; Weihua Ou ; Linghui Liu **All Authors**

10	605
Cites in Papers	Full Text Views

Identification of recaptured photographs on LCD screens

Publisher: IEEE Cite This PDF

Hong Cao ; Alex C. Kot **All Authors**

47	4	814
Cites in Papers	Cites in Patents	Full Text Views

IOActive.

Problems with Video Replay Attacks

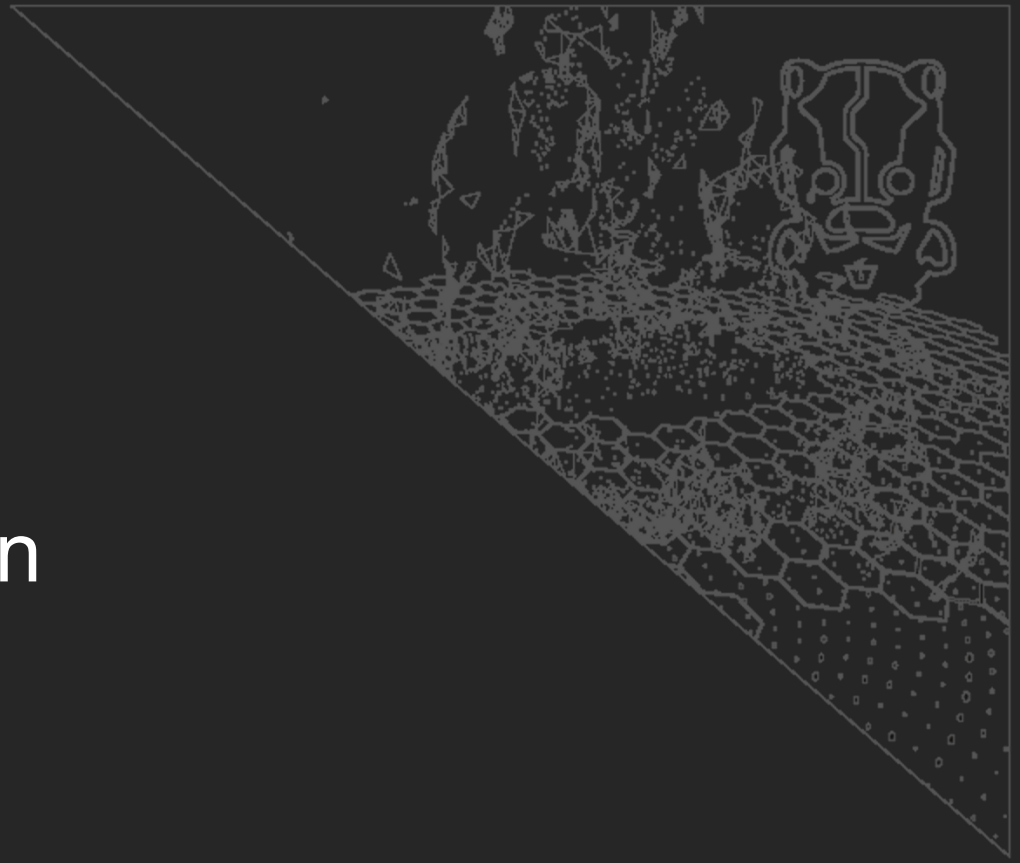
- Loss of quality
  - Color spaces (HSV/HSL/HSI)
  - Focus/blur
  - RGB channels & grayscale
- Presence of display/monitor features
  - Glare
  - Reflections
  - Chromacity
  - Brightness
- And more…

Ideas & Objectives

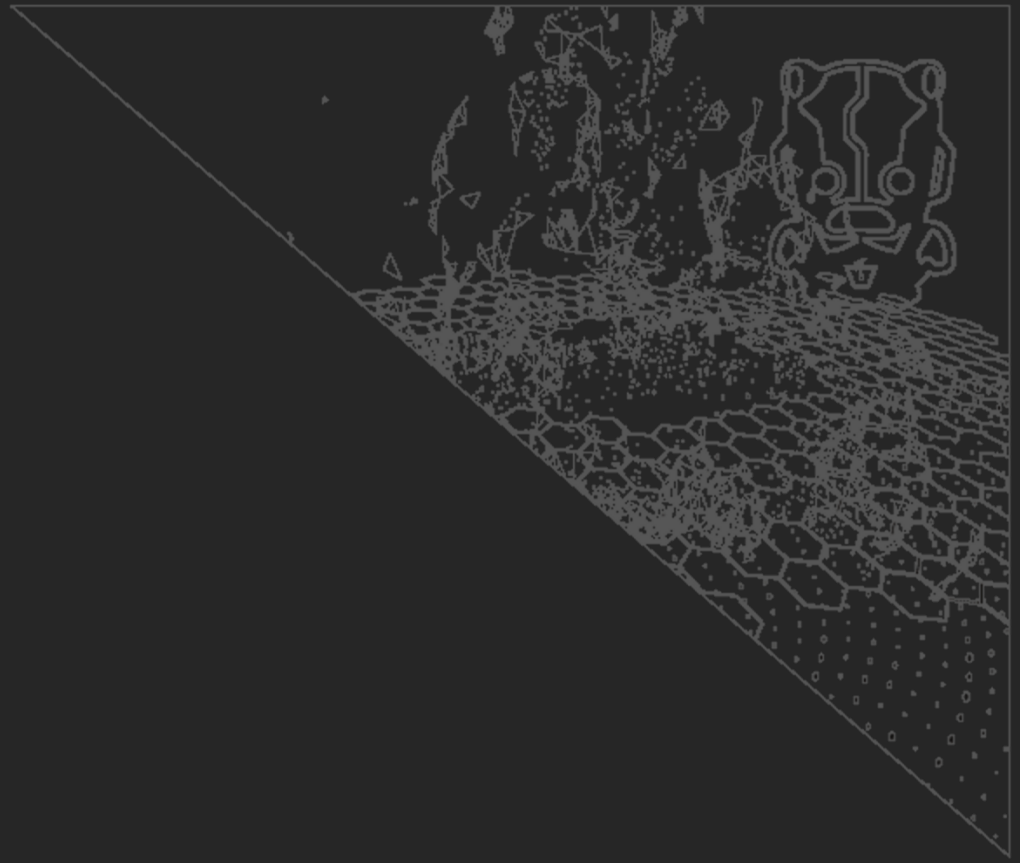
- *Can we trick the sensor/camera by directly feeding the source (video/image) instead of “presenting” them via display/monitor?*

- *Can this be accomplished using free, time-saving, and resource-conserving methods, which work on multiple platforms?*

The (Simple) Solution

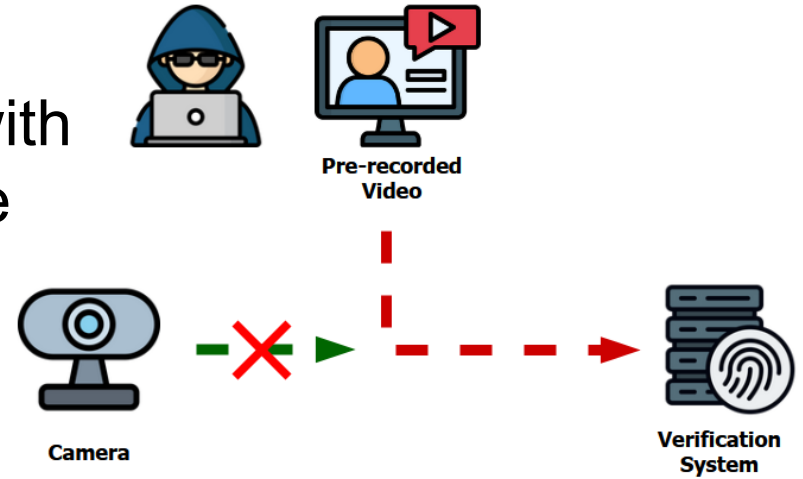


Video Injection



Video Injection

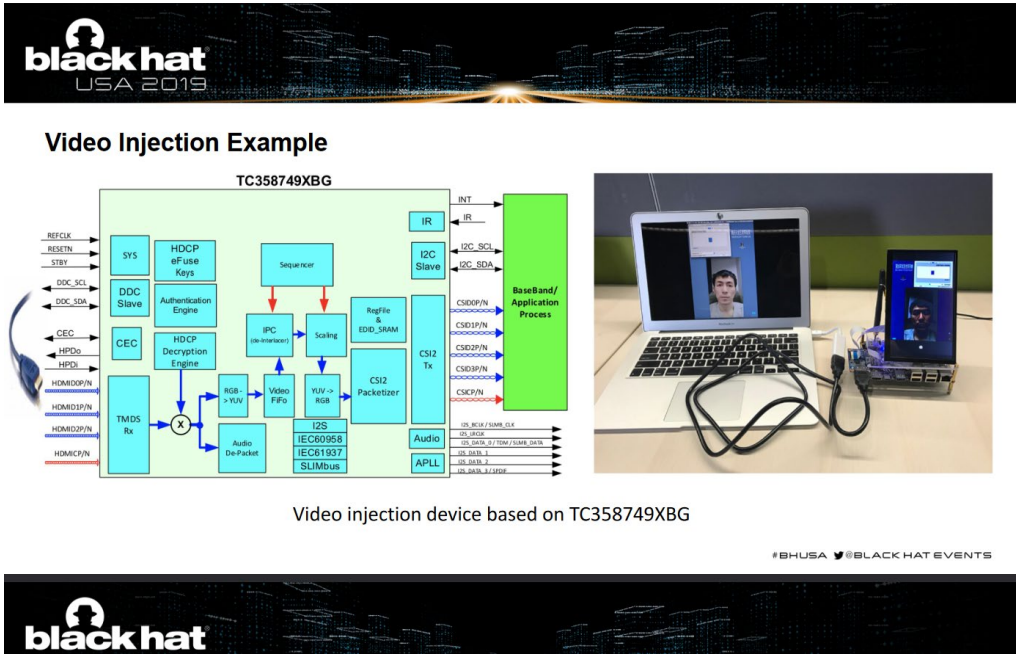
- An attack where the threat actor “feeds” or “injects” the source video (pre-recorded, AI-generated, fake/edited, etc.), instead of a live video feed, directly to the verification system.
- It helps address the problem with Video Replay Attacks since the injected video is bit-for-bit a replica of the source video and of high quality.



IOActive.

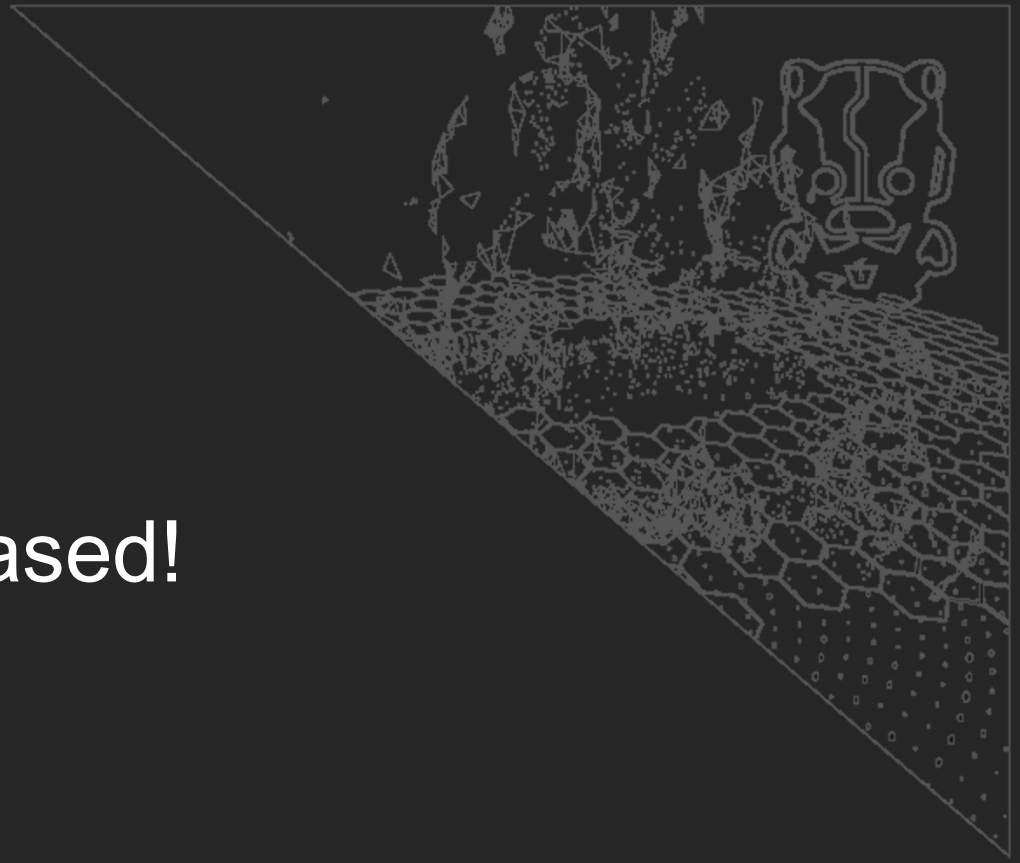
Hardware-based (previous research)

- Presented at Black Hat 2019.
- Effective; however, the setup is costly and resource-intensive.



IOActive.

Let’s Go Software-based!



Virtual Camera

- A software-based (hence, “virtual”) camera that mimics what a physical camera does.
- It allows users to select different sources (images, videos, audio, scenes, etc.) and feed them to other applications, such as video conferencing, video chats, live streaming, etc.
- Useful if the computer does not have a physical webcam.
- Low-cost *(most of them are free)* and easy to set up.

Virtual Camera Providers

IOActive.

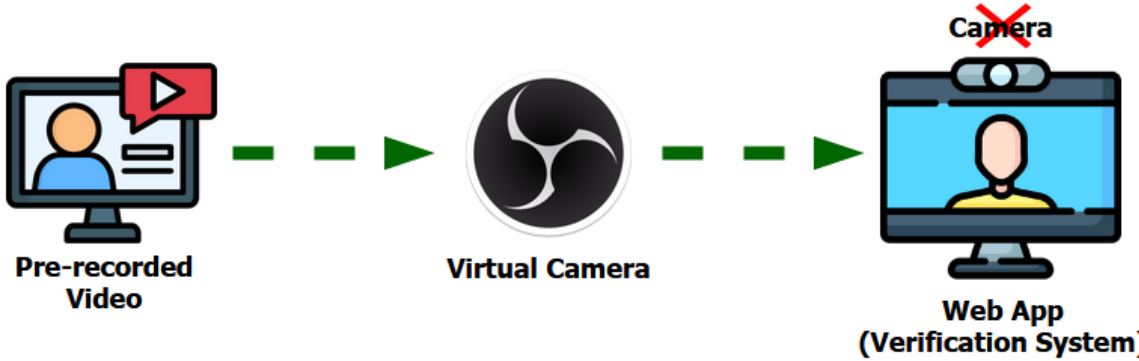
Virtual Camera Attack Methods

- Desktop
- Mobile emulators
- Physical mobile device

Method #1: Via Desktop

Via Desktop

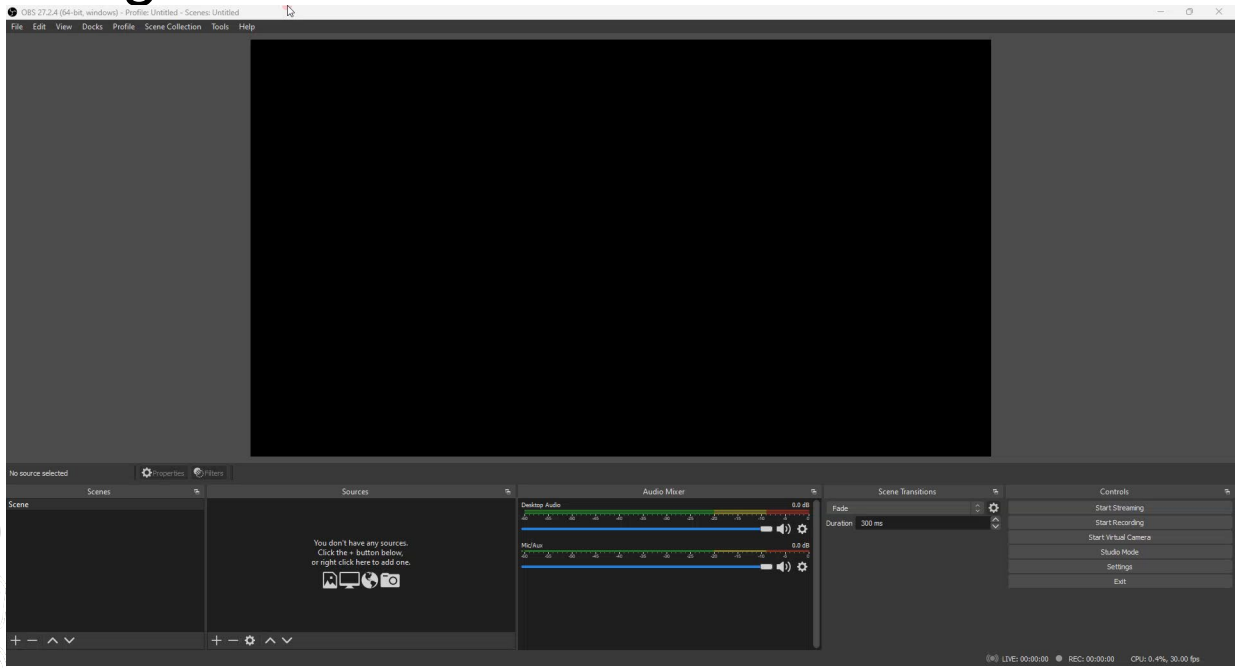
- The user is using a desktop to perform facial liveness verification through the provider's web application.
- Instead of using the desktop's physical camera, a pre-recorded video is fed to the virtual camera.



IOActive.

Via Desktop (Sample Setup)

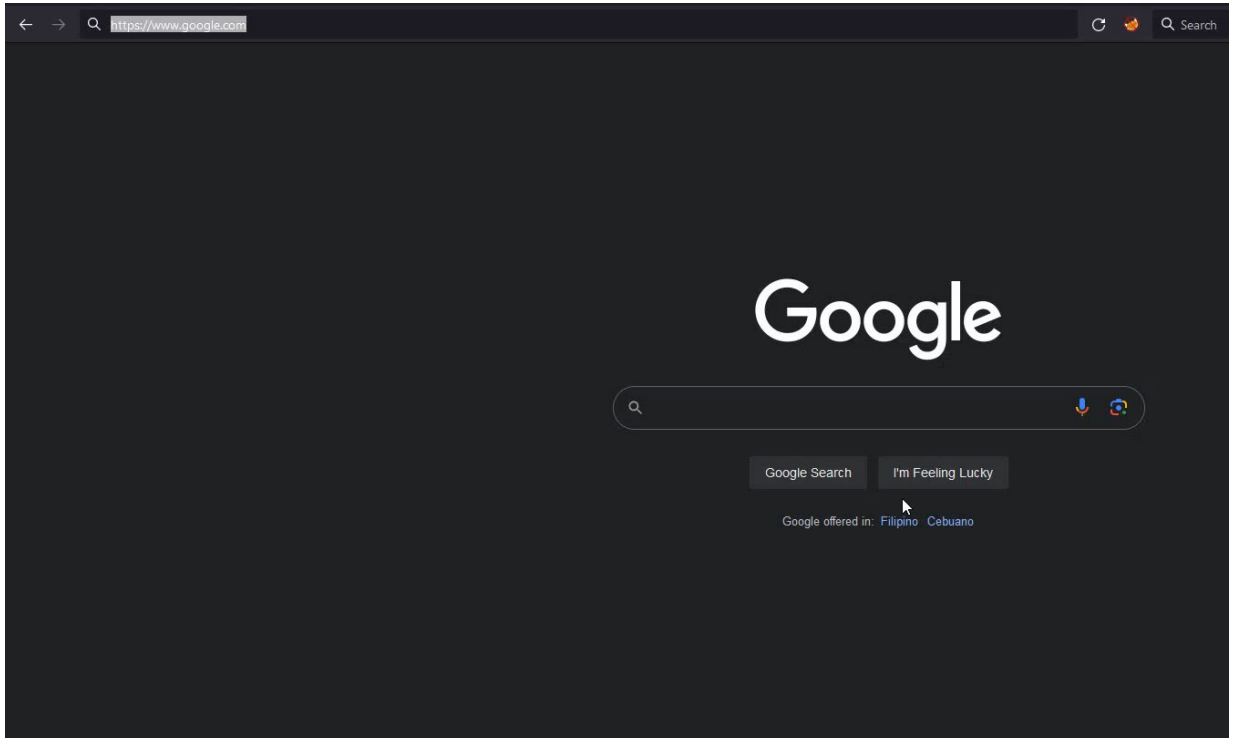
- Using OBS Studio Virtual Camera.



Credits: video from https://www.pexels.com/video/woman-wearing-face-mask-having-a-virtual-meeting-at-the-office-8135796/

IOActive.

Via Desktop (Sample Verification)

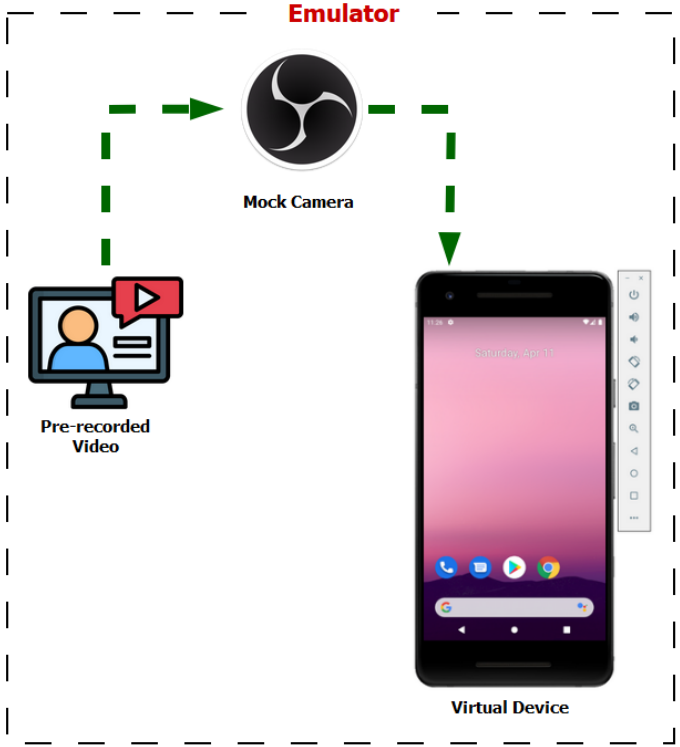


IOActive.

Method #2: Via Mobile Emulators

Via Mobile Emulators

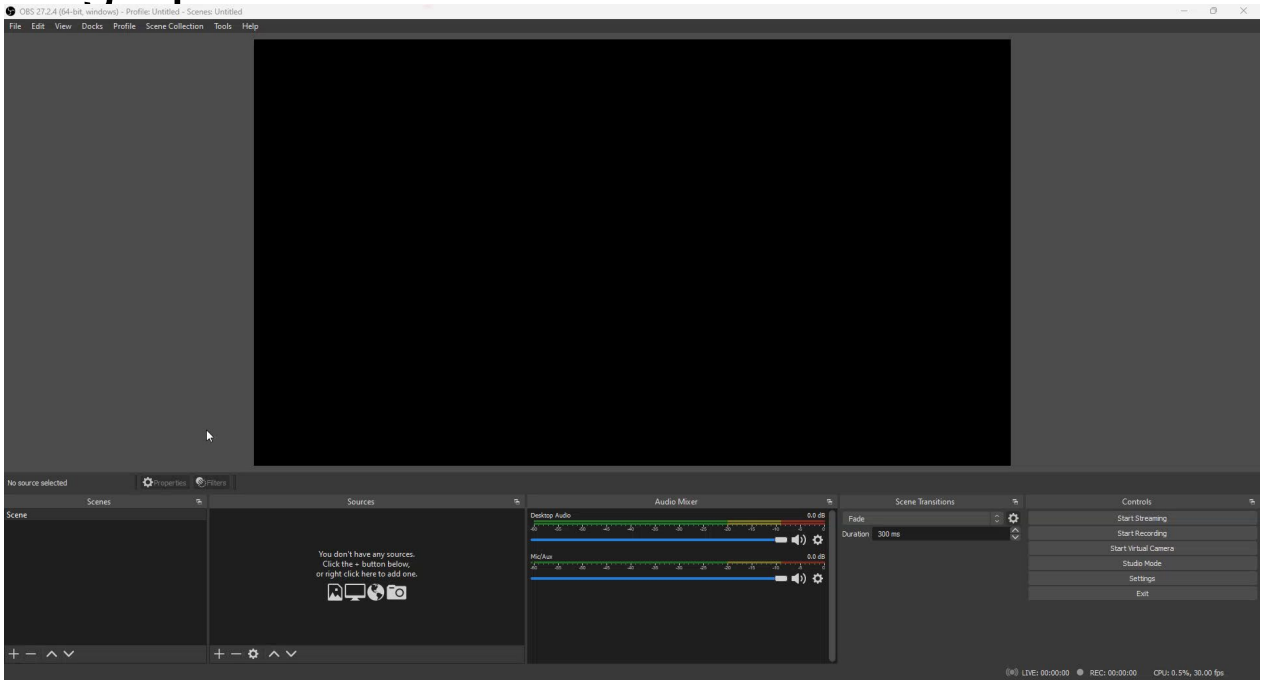
- The user is performing the verification using the provider's mobile application.
- An emulator is used to mimic a real phone's functionalities.
- Pre-recorded videos are fed into the emulator's mock camera.



IOActive.

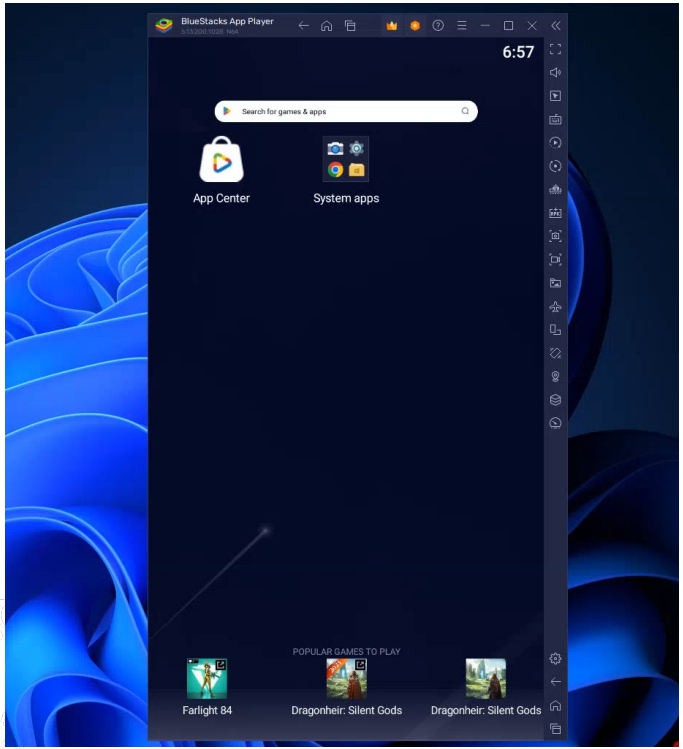
Via Mobile Emulators (Sample Setup)

- Setting up OBS Studio Virtual Camera.



Credits: video from https://www.pexels.com/video/two-teenagers-wearing-eyeglasses-looking-at-camera-6272052/

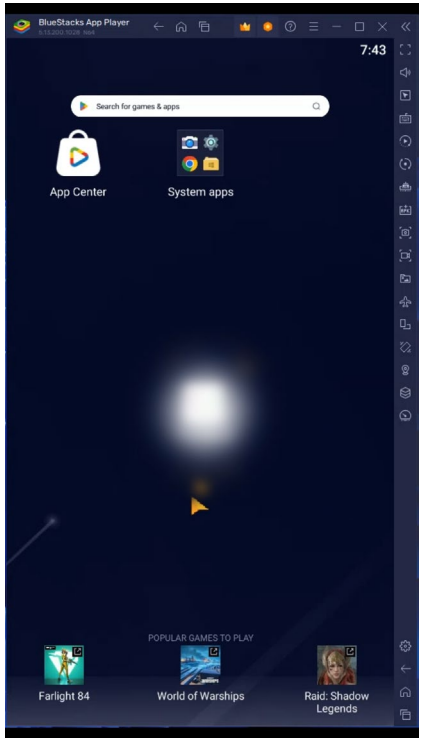
Via Mobile Emulators (Sample Setup)



- Setting up BlueStacks as the emulator.
- Selecting OBS Virtual Camera as the device camera for Bluestacks.

IOActive.

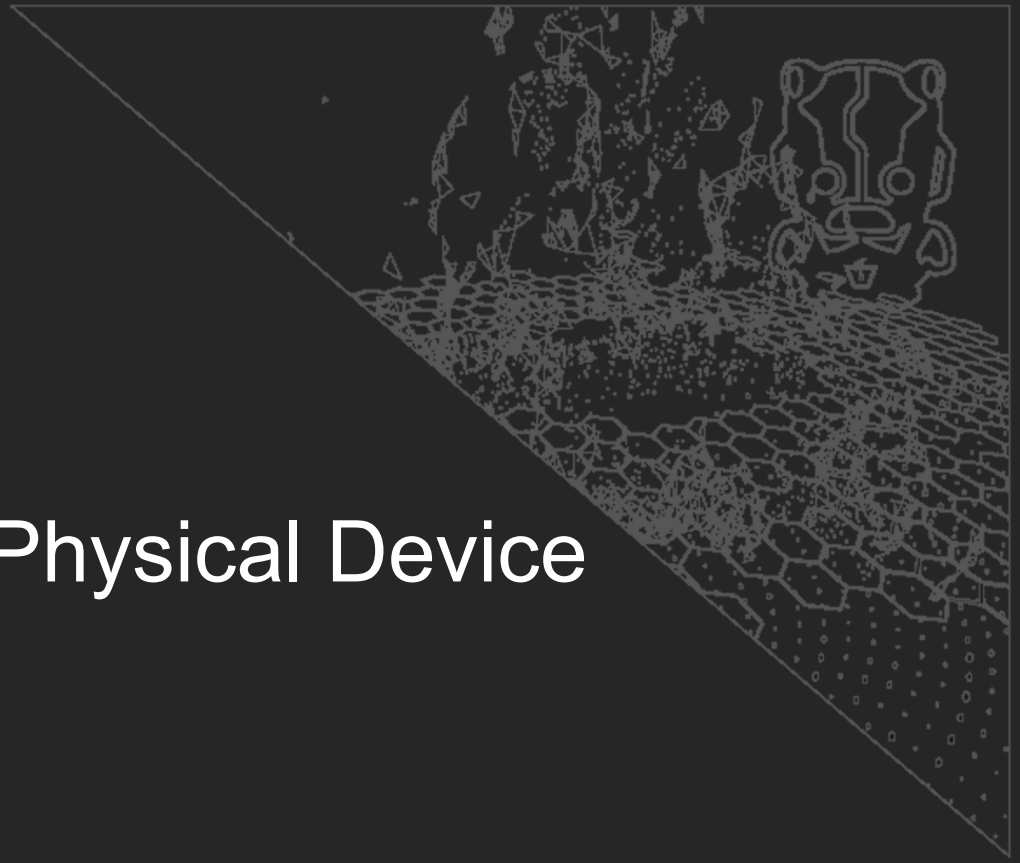
Via Mobile Emulators (Sample Verification)



Problems with Emulators

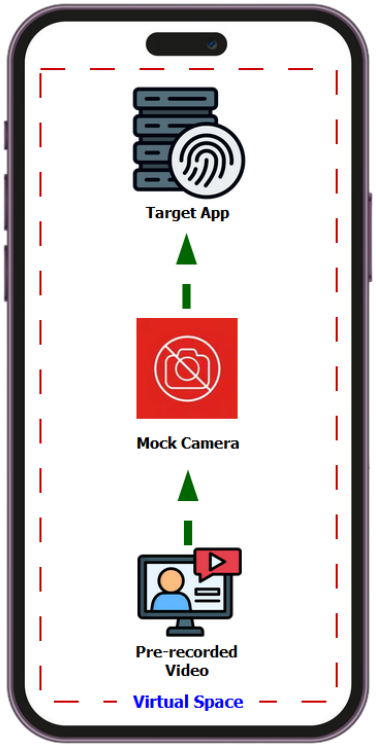
- What if the mobile application has an emulator detection to prevent it from running in an emulator?
- What if anti-tampering techniques are implemented to prevent bypassing/defeating the anti-emulation techniques?
- User's only option is to carry out the facial verification through the use of a real/physical mobile device.

Method #3: Using a Physical Device



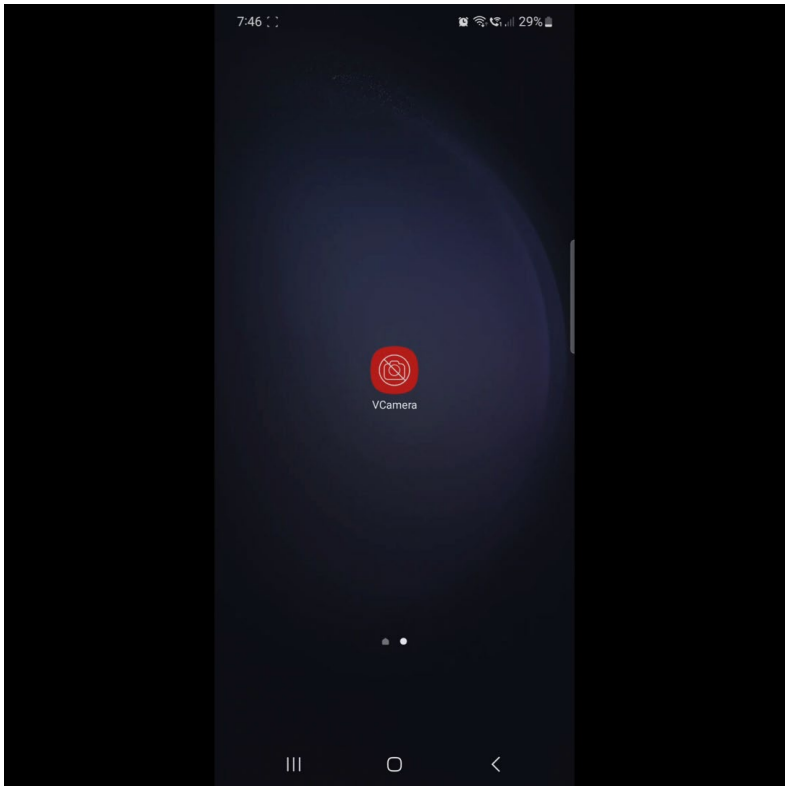
Using a Physical Device (Android)

- The user is carrying out the facial verification using the provider's mobile application, installed in the user's real/physical mobile device.
- The application is installed in a virtual environment running in the Android phone.
- Pre-recorded videos are fed into the VM's mock camera.



IOActive.

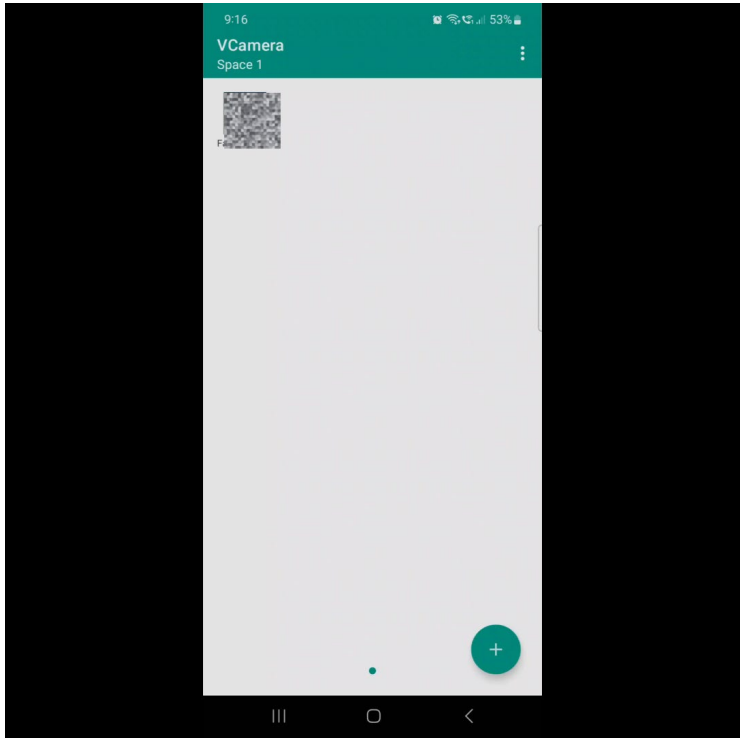
Using a Physical Device (Sample Setup)



- Setting up Virtual Camera:Live Assist

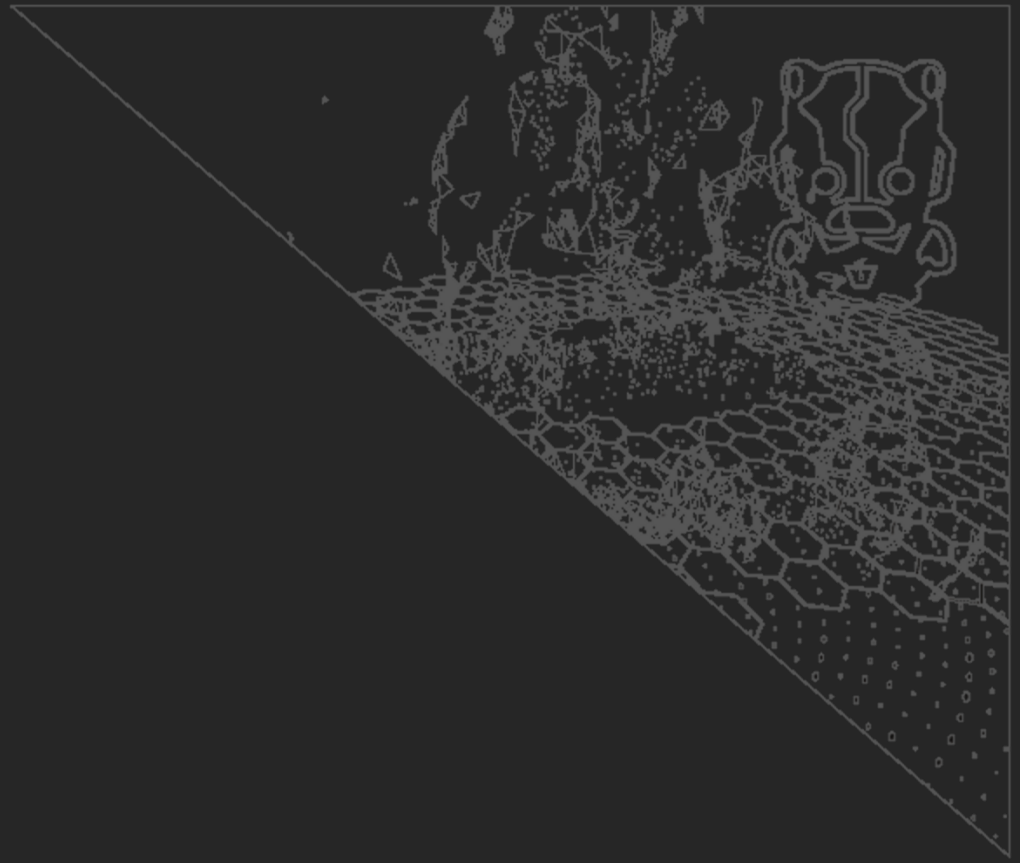
Credits: video from https://www.pexels.com/video/man-in-gray-coat-happily-looking-at-the-camera-5989757/

Using a Physical Device (Sample Verification)



IOActive.

Pros & Cons



Pros & Cons

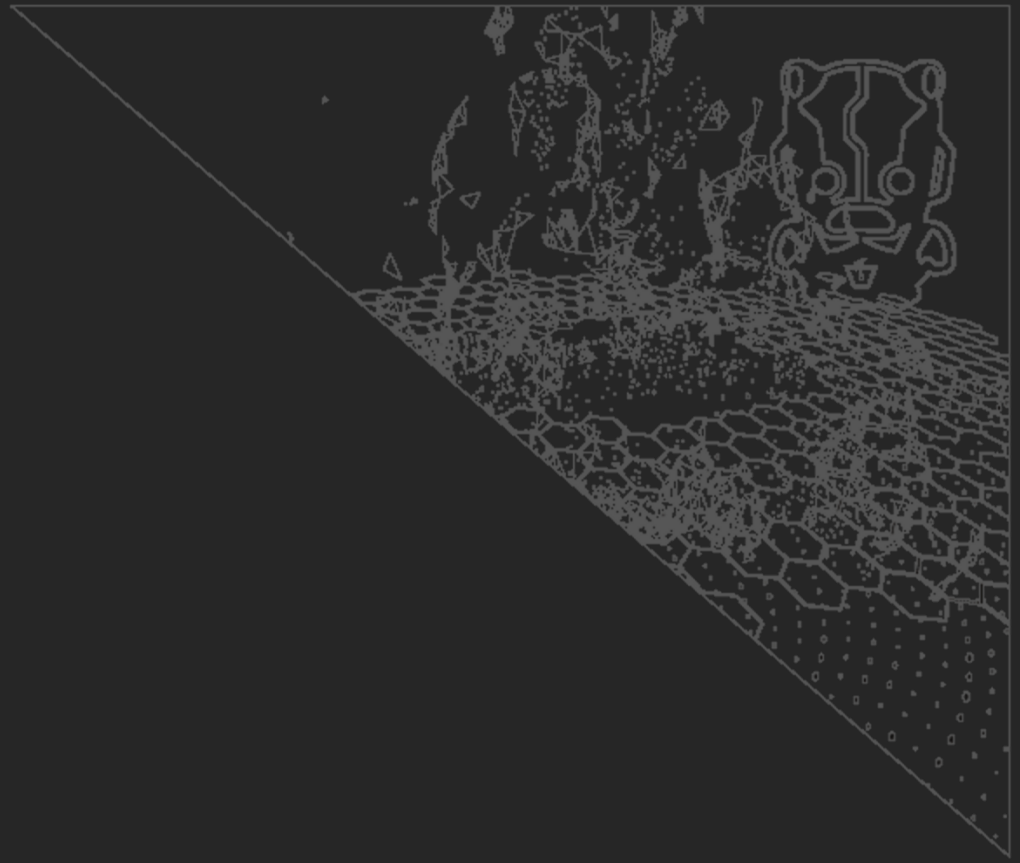
Pros:

- Simple and easy to configure
- Low-cost *(mostly free and open-source)*
- Tools are readily-available
- Not time and resource-intensive *(compared with using hardware modules, generating deepfakes, manufacturing hyper-realistic silicon masks, etc.)*

Cons:

- No assurance that the setup will work on the first try *(crucial if the verification system logs every verification attempt and locks the users)*
- Might not work on all verification platforms

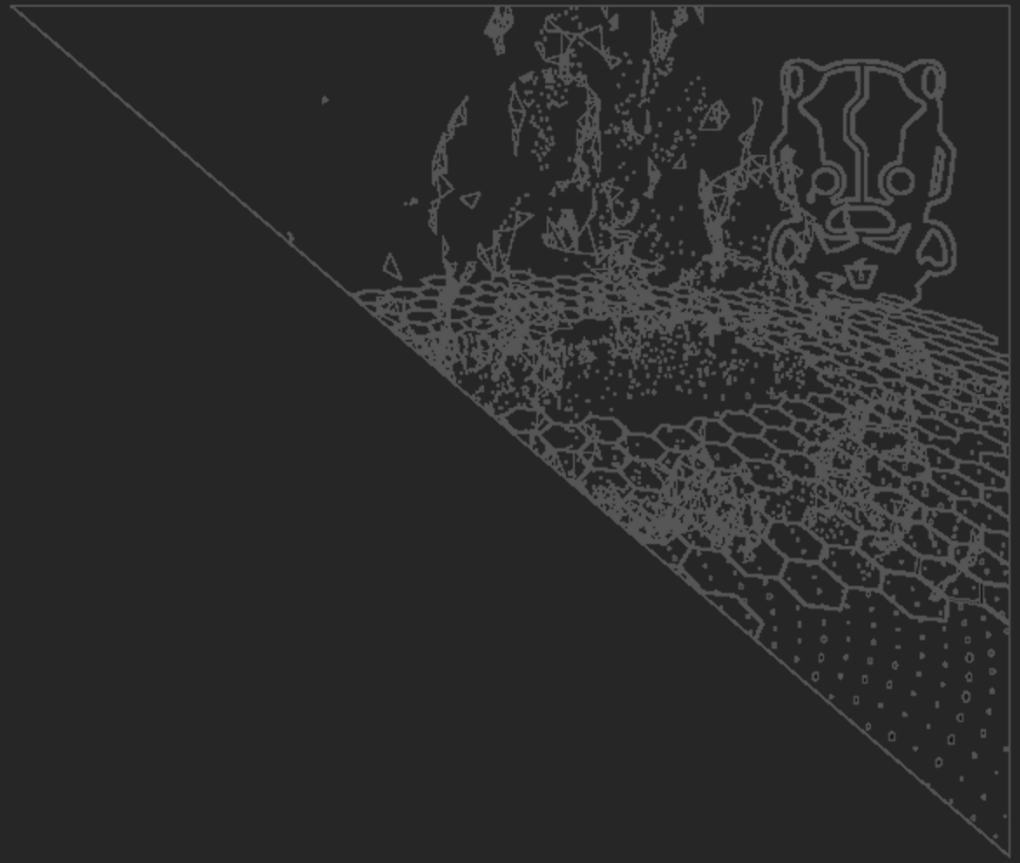
Threats



Threats

- Nowadays, identity theft is easier as fraudsters can effortlessly obtain pre-recorded videos of their targets via:
  - Social media platforms (e.g., TikTok)
  - Vlogs
  - Video calls or virtual meetings
  - Or any video where a user's face is always fronted

The Way Forward



Detect Virtual Cameras

- Differentiate a real/physical camera vs a virtual one
  - Identifying the device's name *(beware as some vcams have a feature to change its name).*
  - Checking the list of supported resolutions *(vcams often have different supported resolutions than physical cameras).*
  - Examining the camera's API functions.

Protect Against Emulation

- Detect the presence of emulated environments and prevent applications from running in such environments:
  - Emulator artifacts *(e.g., specific files, system properties, emulator-specific packages, etc.)*
  - Hardware characteristics and identifiers *(e.g., device's model, manufacturer, sensor data, etc.)*
  - Device performance *(e.g., CPU speed, memory availability, graphics capabilities, etc.)*
  - Network environments *(e.g., MAC addresses, TTL, etc.)*

Secure & Analyze the Input Video

- Ensure authenticity and fidelity of the input data *(e.g., remote image attestation):*
  - A private key is used to sign the hash of the captured media. Once the media has been captured, its hash will be signed with a private key.
  - Once submitted, the media's hash will be recalculated on the server side with the same hash function and then signed with a public key.
  - The media is considered authentic and tamper-free if the signed hash matches the hash of the received media.
- Improve AI to better detect anomalies in video feeds *(e.g., looping video, deepfakes, digital filters or enhancements, etc.)*

Employ a Layered Approach

- Do not rely on only one form of identity verification:
  - Facial liveness + document/ID validation with face-matching
  - Comparison of live selfie with a reference selfie
- Monitor (failed) verification attempts *(normally, pre-recorded videos do not work on the first try)*.

Keep Improving

- Employ a better active liveness detection wherein users are required to perform a sequence of **true random** movements.
- For highly sensitive applications, consider a manual, live, and/or facilitated identity verification *(e.g., a proctor would require users to do certain tasks).*

Thank you!