Trustworthy Attestation of Untrusted Sensors

Junia Valente

Email: juniavalente@utdallas.edu

Abstract

Critical infrastructures are monitored and controlled by a wide variety of sensors and controllers. Unfortunately, most of these devices interacting with the physical world are fragile to security incidents.

One particular technology that can help us improve their trustworthiness is *attestation*—a protocol where a verifier sends a random challenge to a device and the device replies with a response to prove its integrity (Fig. 1). However, the need to modify the device software or lack of hardware (e.g., TPM) have limited the use of attestation on existing devices.

Our contribution: we propose a new attestation protocol to detect replay-attacks on sensors that continuously monitor a physical area. The breakthrough aspect of our approach is that we do not send the attestation challenge to the device itself, instead we modify the physical area the device is sensing and verify that the desired changes are reflected in the sensor readings. We evaluate our approach for IP cameras used in surveillance applications.

Introduction

Problem Description:

- Sensing devices that interact with the physical world are extremely fragile to security incidents
- Once secret keys of a device are compromised, the attacker can bypass traditional integrity mechanisms

Attestation to Improve Device Trustworthiness:

- Attestation helps detect unauthorized changes to devices
- Drawbacks have limited their application in the field

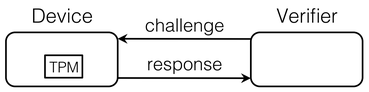

Figure 1: Generic attestation protocol.

Approach and Uniqueness

Our Proposed Solution is motivated by attestation; however, the novel aspect is that we do not send the attestation challenge to the device itself. Instead, we modify the physical area the device is sensing and verify that the desired changes reflect in the sensor readings.

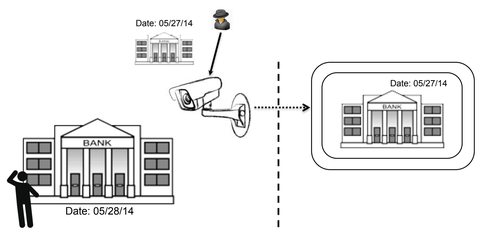

Figure 2: An intuitive illustration of our approach

Proposed Solution Overview

Proposed Attestation for Sensing Devices:

- The verifier sends a random challenge as input to the prover by modifying the physical area the prover is sensing (Fig. 3)
- The response will show up in the output of the prover even if the prover does not know anything about the challenge sent
- If the correct response does not show up in the prover output, then we know the prover is not reporting correct readings

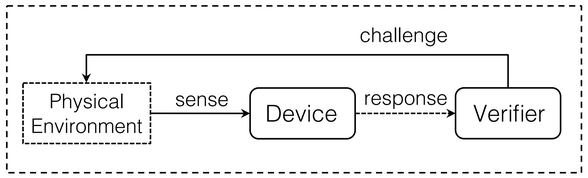

Figure 3: The verifier modifies the physical environment that the sensor (labeled as the 'device') is sensing. The verifier then verify that the sensor readings (labeled as 'response') include the injected 'challenge'.

Approach Evaluation:

- We evaluate our approach for surveillance cameras
- Security cameras are used in sensitive settings and are becoming attractive to attackers

Adversary Model:

The attacker can launch replay attacks by using old video footage. Our goal is to minimize the changes the attacker can use previously recorded footage and present it as new.

Research Challenges:

- What is the *practicability* of our proposed attestation?
- How can we inject *verifiable evidence* in the camera video feed for the purpose of attestation?
- How can we continuously *modify* the physical world in a random way so the modifications can be *communicated* to the verifier?

Implementation Details

System Design:

Our goal is to detect a compromised camera in the presence of an attacker, i.e., a camera reporting incorrect or old data.

We add two new devices: a **display** and a **verifier**.

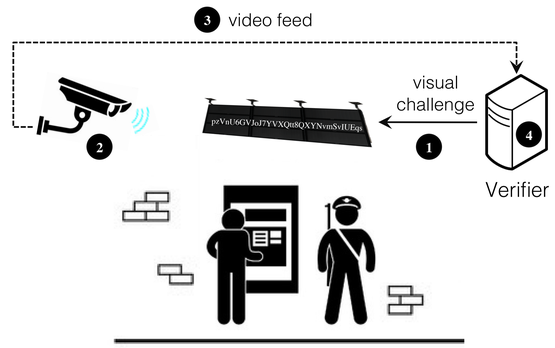

Figure 4: (1) A visual challenge is sent to a display, (2) the camera captures an image which includes the display, (3) the video feed is sent to verifier, (4) if the verifier confirms the challenge was captured in the video just received, it gains confidence that the camera is transmitting fresh footage.

We Propose Two Visual Challenges with enough randomness to prevent replay attacks on cameras: plain text (e.g., Fig. 4) and QR code (e.g., Fig. 5).

- Plain text can be recognized with optical character recognition (OCR) such as **`tesseract`** [4]
- QR code can be decoded via popular barcode readers such as **`zbar`** [7]

Verification Process:

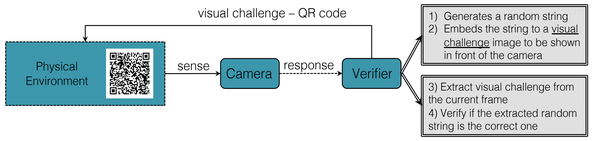

Figure 5: Verifier generates a random string and either encodes it into a QR code or displays it as-is in the monitor. The verifier use QR reader or OCR engine to extract the random string from the image before verification.

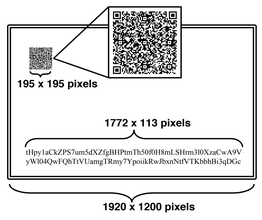

Figure 6: QR code vs. Plain Text.

Figure 7: Lab Setup for Experiments.

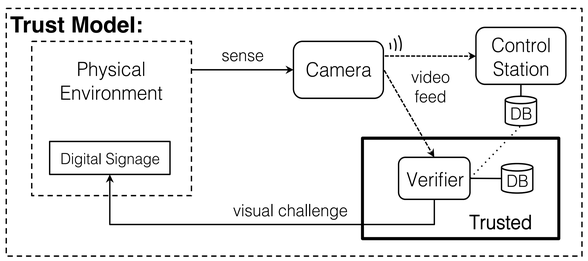

Figure 8: The only trusted entities in our system are the verifier, which sends visual challenges to the display, and the database that stores the history of sent challenges for offline forensics purposes.

Results and Conclusions

Our results show that displaying a continuous stream of visual challenges on a signage is an effective way to communicate verifiable evidences between the verifier and camera.

In our upcoming ACSAC'15 paper [5], we propose attack detection statistics for each visual challenge, and study their performance under normal conditions (without attack) and against a variety of adversaries.

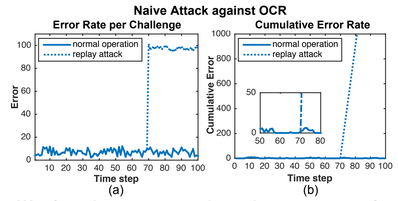

Figure 9: We found our approach to be secure against replay-attacks even when the camera footage might not look suspicious to security guards [5]. Fig 8 (a): anomaly score per image. Fig 8 (b): cumulative anomaly score over time (i.e., the CUSUM statistic).

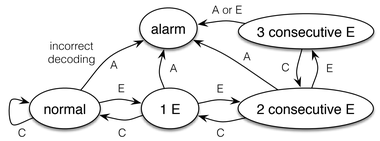

Figure 10: Anomaly detection for QR code visual challenges. Legend: C = correct decoding, A = decoding of a value different to the challenge, and E = cannot decode QR code in the current frame.

Related Works

There has been attention on ensuring camera systems secure video streams in the cloud and in transmission (by signing [2] or encrypting [3] the stream before sending to storage). However, these approaches assume keys have not been compromised.

Research on the security of sensors includes those that look at the development of *trusted sensors* to ensure trustworthy sensor readings. Proposed approaches [6] make it difficult for a sensor to 'lie' or fabricate malicious readings because they assume the presence of trusted computing technology [1].

Acknowledgements

This research was conducted under the supervision of Dr. Alvaro Cárdenas. This work was supported in part by NIST under award 70NANB14H236 from the U.S. Department of Commerce.

References

[1] D. Challener, K. Yoder, R. Catherman, D. Safford, and L. Van Doorn. A practical guide to trusted computing. IBM Press, 2008.

[2] G. Friedman. The trustworthy digital camera: restoring credibility to the photographic image. IEEE Trans. Consum. Electron, 39(4):905–910, 1993

[3] R. Hummen, M. Henze, D. Catrein, and K. Wehrle. A cloud design for user-controlled storage and processing of sensor data. In CloudCom, 2012.

[4] Tesseract Open Source OCR Engine. https://github.com/tesseract-ocr/.

[5] J. Valente and A. Cárdenas. IOT: Using Visual Challenges to Verify the Integrity of Security Cameras. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC'15)*, 2015 (forthcoming).

[6] T. Winkler and B. Rinner. Securing embedded smart cameras with trusted computing. EURASIP Journal on Wireless Comm. & Netw., 2011.

[7] ZBar bar code reader. http://zbar.sourceforge.net/.