Towards the Security of Motion Detection-based Video Surveillance on IoT Devices

Xianglong Feng
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
xianglong@huskers.unl.edu

Mengmei Ye
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
mye@huskers.unl.edu

Viswanathan Swaminathan
Adobe Research
San Jose, CA 95110
vishy@adobe.com

Sheng Wei
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
shengwei@unl.edu

ABSTRACT

Video surveillance enabled by Internet of Things (IoT) devices, such as smart cameras, has become a popular set of applications recently with the trend of adopting IoT in multimedia signal processing and smart home use cases. Despite its intelligence and convenience, the video motion detection module deployed on the IoT devices poses security challenges due to the sensitive nature of the captured surveillance video and the motion detection operation. In this paper, we investigate the security vulnerabilities of IoT video surveillance from the hardware system point of view. We first develop a proof-of-concept prototype demonstrating video replay attacks, in which the compromised surveillance device hides the chosen suspicious motion by overwriting the corresponding frames with pre-recorded normal frames under the control of the attacker. To address the security concerns, we develop a hardware-based IoT security framework that creates a trusted execution environment and physically isolates the security sensitive components, such as the motion detection module, from the rest of the system. We implement the security framework on an ARM system on chip (SoC). Our evaluations on the real hardware reveal superior security and low performance/power overhead in IoT video surveillance applications.

1 INTRODUCTION

Smart surveillance cameras, such as Nest Cam [3] and Ring video doorbell [4], have gained great popularities recently with the rapidly growing trend of deploying IoT devices in smart home environments. Different from the traditional surveillance flow of video capture, delivery, and playback, the Internet of Things (IoT) surveillance camera is "smart" in that it can process the captured raw video and, in particular, automatically detect the moving objects in the video. Such a motion detection feature can save a large amount of network bandwidth by delivering only the video with motion (i.e., containing suspicious behavior) or simply an alert to the receiver's end. Also, it significantly reduces the required manpower in security surveillance, as the camera can automatically report security incidents based on motion.

While conducting the capture and processing of the surveillance video, the security and privacy of the smart camera become critical. First, the subject under surveillance is typically a private residence, which is privacy sensitive. Second, the video is used for security surveillance, where the attackers have a natural incentive to compromise the system component that handles video processing, such as the motion detection module, in order to bypass the security screening. In both cases, the key algorithms and libraries deployed on the smart camera must be well protected to prevent security or privacy breaches. Furthermore, as a time sensitive application, video surveillance requires near real-time speed and latency in the video processing and delivery, which is challenging due to the computational complexity involved.

The general state-of-the-art solution to address the video security and privacy challenges is to encrypt the video content throughout the entire video pipeline, which has been widely adopted in entertainment video streaming [31][32][34]. However, we observe that the encryption-based security mechanism does not apply to the protection of IoT surveillance videos. In particular, the attackers are able to issue replay attacks that pre-record non-motion video frames to replace the ones with motion and thus bypass the video surveillance. To the best of our knowledge, there has been neither investigations nor solutions towards this security aspect of IoT video surveillance.

In this paper, we first develop a smart video surveillance prototype for the purpose of security analysis and development using a programmable system on chip (SoC). On the SoC, we develop a motion detection module using a simplified Gaussian mixture model (GMM) based algorithm, which is deployed in the programmable logic part of the SoC representing a third-party video intellectual property (IP) core. Then, on top of the video surveillance prototype, we develop a proof-of-concept threat model demonstrating video replay attacks, in which a malicious software compromises the surveillance IoT device by hiding the chosen motions via overwriting the motion frames with pre-recorded normal frames. In this way, the attacker is able to gain full control over the motion detection module of the surveillance system and manipulate the

ThematicWorkshops'17, October 23–27, 2017, Mountain View, CA, USA
© 2017 ACM. ISBN 978-1-4503-5416-5/17/10…$15.00
DOI: http://dx.doi.org/10.1145/3126686.3126713

motion-based alerts, which compromises the security of the surveillance application.

To address the security concerns, we develop a hardware-based IoT security framework that creates a trusted execution environment and physically isolates the security sensitive components, such as the motion detection module, from the rest of the system. We implement the security framework leveraging the TrustZone technology [1] provided by the ARM processor, which is the most popular embedded processor used in mobile and IoT devices. Our evaluation on an ARM SoC proves the enhanced security and the minimum performance overhead brought by the proposed multimedia IoT security framework.

To summarize, our technical contributions in this paper include the following:

- We develop a surveillance video prototype using hardware motion detection on a programmable SoC;
- We showcase a replay threat model targeting the motion detection module in IoT video surveillance; and
- We develop a hardware isolation-based security framework that isolates the security sensitive video pipeline from the attacks and introduces minimum performance and power overhead.

To the best of our knowledge, our work is the first investigating the security of IoT video surveillance from the hardware system point of view. By presenting this work we aim to motivate a new thread of research leveraging hardware-based techniques to address IoT multimedia security challenges.

The remainder of the paper is organized as follows. In Section 2, we introduce the prototype video surveillance system that we develop in the lab environment for security analysis. Section 3 focuses on the potential threat model we study in this work, namely the replay attack. Based on the threat model, in Section 4, we introduce our hardware isolation-based security framework that serves as an effective countermeasure for the proposed replay attack. Section 5 presents our experimental results for both the threat model and the countermeasure. Section 6 summarizes the closely related prior work. Section 7 discusses additional threat models we identified upon accomplishing this work, which motivate our future work on IoT multimedia security. Finally, Section 8 concludes the paper.

2 SURVEILLANCE VIDEO SYSTEM ON CHIP

We develop a prototype of IoT surveillance video system on a programmable SoC in order to investigate the threat models and countermeasures in our lab environment. In this section, we introduce the overall system architecture of the prototype system, as well as the detailed design of the motion detection module, which is the target of the security analysis and development.

2.1 System Architecture

Our prototype system is built on a small embedded SoC representing a commonly used IoT device, as shown in Figure 1. The core component of the video surveillance system is a Xilinx Zynq SoC ZC702, which obtains the source surveillance video from the laptop computer (i.e., emulating a secure camera), processes the video for motion detection, and outputs the motion detection results to the monitor (i.e., emulating a security alert of objects in motion).

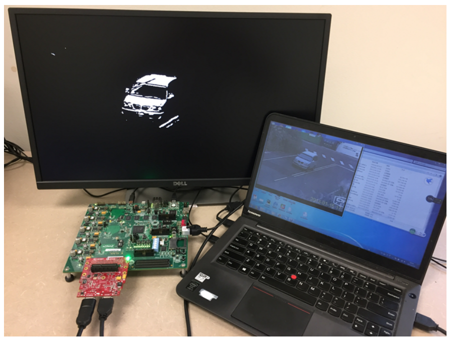


Figure 1: Hardware setup of the surveillance video system.

Figure 2 illustrates the detailed internal architecture and workflow of the SoC, which consists of two major system components, namely the processing system (PS) and the programmable logic (PL). The PS is driven by the application processor (e.g., the ARM processor) and involves user applications based on general purpose computing conducted by the CPU. The PL involves programmable hardware components (e.g., FPGA) and other peripheral devices aiming to accelerate domain specific computations, such as video processing. In the prototype of video surveillance, we deploy the motion detection module in the PL part to gain superior performance from hardware acceleration. The PS and the PL are connected by the AXI interconnect, which can manage the communications between the two components.

With the aforementioned system architecture, our motion detection-based video surveillance workflow is the following. First, the HDMI input module obtains the input video data from the video source and decodes it into the YCrCb format, which is stored in the DDR memory through video direct memory access (VDMA). Then, the motion detection module loads the video data from the DDR memory and conducts motion detection operations (discussed in details in Section 2.2). Finally, the motion detection module generates black and white output frames as the results, in which the motion part is labeled as white and the non-motion part is labeled as black.

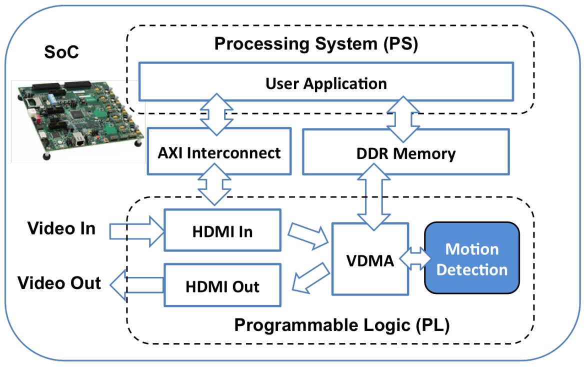


Figure 2: Workflow of the surveillance video system.

2.2 Motion Detection Algorithm

In this subsection, we discuss the details of our motion detection algorithm that is deployed in the motion detection module in the PL part of the SoC. The technical objective of motion detection is to differentiate the foreground and background in the video and detect the motion in the foreground. A straightforward idea to achieve this goal is to subtract a frame from its preceding frame. The subtraction results would contain zeros and non-zeros, which represent the static background and the objects in motion, respectively. Such an algorithm is simple and fast, as the only required computation is the subtraction of two frames. Therefore, it is very easy to achieve real-time performance with this algorithm, even if there is no hardware acceleration. However, the algorithm assumes that the background of the video is always static, while in most use cases the luminance of the surroundings is varying all the time considering the environmental factors like wind, cloud, rain, and the position of the sun. Consequently, the accuracy of the algorithm cannot be guaranteed. In order to achieve a better accuracy, the "preceding frame" should be updated regularly and, at the same time, it should not include the foreground objects in motion.

To address the aforementioned accuracy problem, researchers have proposed Gaussian Mixture Model (GMM) based approach [29]. The key observation in GMM is that the pixel values in the background often fit in a Gaussian distribution. In a more complicated environment, multiple Gaussian models can be adopted to present the background and thus the name Gaussian Mixture Model. As presented in Figure 3, the probability of a given pixel belonging to the background can be calculated using the Gaussian models (i.e., variations of the pixel values in the background). Note that the Gaussian models are dynamic and can be updated using the detection results in the real time.

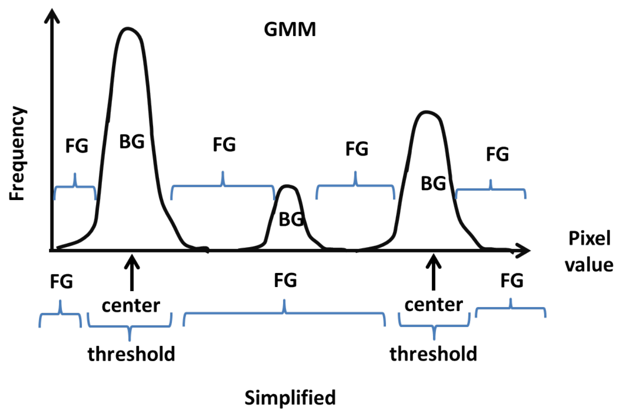


Figure 3: Gaussian mixture model and our simplified model for motion detection (FG - Foreground; BG - Background).

The GMM-based algorithm has very good accuracy and is less influenced by the surrounding luminance variance. However, there are two challenges in applying the GMM algorithm in the video surveillance scenario. First, the complexity of computation is huge since the algorithm must calculate the value of the Gaussian function for each pixel in a frame. Second, the algorithm needs relatively large memory space to store a GMM for each pixel. Therefore, a balanced algorithm is required for embedded systems that are resource constrained.

In our prototype system, as shown in Figure 3, we develop a simplified mixture model algorithm where we replace the Gaussian model with a center value and a threshold, which is illustrated below the x-axis. Given a pixel, we subtract it from the center value and compare the results with the threshold to determine if the pixel belongs to the background. As a result, our algorithm does not require the Gaussian models and thus stores fewer parameters. Furthermore, we implement the motion detection algorithm in the PL part of the SoC, which further benefits from the performance acceleration brought by the FPGA. All these design and implementation strategies combined ensure the superior performance of the motion detection module for the real-time video surveillance application.

3 THREAT MODEL: REPLAY ATTACK

The video surveillance system is typically deployed as an IoT smart camera, which helps monitor a security sensitive area, such as private residence. The smart camera is capable of monitoring the video of the area under surveillance for suspicious motion and generating a security alert if there is any detected. However, being deployed as an IoT device connected to the Internet, the smart camera is vulnerable to security attacks, as the attackers have the incentive to either steal the video stream (i.e., breaking privacy) or falsify the motion data to hide the malicious behavior under surveillance from being captured.

In this paper, we focus on the data falsifying threat and, in particular, replay attack where the compromised camera pre-records non-motion frames to overwrite the frames with motion. In this section, we show that such a replay attack can be implemented in the form of a malicious software embedded in the PS part of the SoC, which can compromise the entire video surveillance system. Figure 4 shows the overall architecture of the replay attack in which the malware has the ability to compromise the DDR memory that hosts the output video frames and thus issue the replay attack.

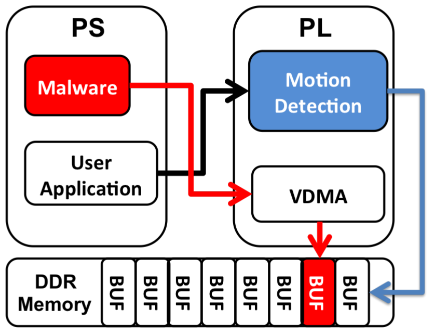


Figure 4: System architecture of the replay attack targeting the surveillance video system.

In particular, the adversary attempts to compromise the software portion (i.e., the PS) of the video surveillance system in order to overwrite the motion frames. Figure 5 describes the software attack flow, which includes five steps.

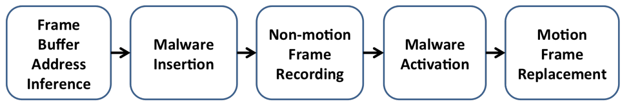


Figure 5: Software attack workflow.

Step 1: Frame Buffer Address Inference. The very first step of the software attack is to infer the buffer address in the memory that hosts the sensitive output frames. An attacker can leverage the ARM compiler to disassemble the video system firmware and infer the key application parameters. Table 1 shows a fraction of the disassembled code in the main function of the software. We observe that there are two memory addresses that appear to indicate video buffers, as shown at lines #103060 and #103070. The attacker can track these candidate addresses to locate the output video frame buffer.

Table 1: Sample of disassembled firmware for the prototype surveillance video system.

103054:	e583200c	str	r2, [r3, #12]	
103058:	e3003040	movw	r3, #64	;0x40
10305c:	e3403011	movt	r3, #17	
103060:	e3a02201	mov	r2, #268435456	;0x10000000
103064:	e5832010	str	r2, [r3, #16]	
103068:	e3003040	movw	r3, #64	;0x40
10306c:	e3403011	movt	r3, #17	
103070:	e3a02202	mov	r2, #536870912	;0x20000000
103074:	e5832018	str	r2, [r3, #24]	
103078:	e3003040	movw	r3, #64	;0x40

Step2: Malware Insertion. With the information from Step 1, the attacker can insert a malware into the software application, either by compromising the software update process or remotely hacking the connected device. The inserted malware is aware of the inferred frame buffer address and the VDMA address. Therefore, it has the ability to overwrite the original output frames.

Step 3: Non-motion Frame Recording. After the malware is inserted into the system, it starts to monitor the status of the video processing pipeline. In particular, the malware examines each output frame and intentionally records those that have no motion into a certain memory space. These non-motion frames are used in the replay attack to overwrite and hide the identified motion frames.

Step 4: Malware Activation. The attacker has full control over the malware and can choose to activate it only when it is necessary to hide the suspicious behavior, while the whole system behaves normal most of the time to bypass security checks. In particular, the activation condition could be a timestamp under the attacker's control or a specific pixel pattern in the input video frame. In our work, we design a time-triggered malware, which is activated at a certain time chosen by the attacker.

Step 5: Motion Frame Replacement. Once the malware is activated, as shown in Figure 4, it issues the replay attack by copying the pre-recorded non-motion frames to the output frame buffer in a random manner. The randomness in the replay ensures that the output video does not represent any noticeable pattern in the background, which can be hidden in random noises that naturally exist in the surveillance video. Therefore, the replayed frames can be hardly distinguished from the normal non-motion scenario.

4 COUNTERMEASURE

In this section, we discuss our new countermeasures to defend against the aforementioned replay attacks on IoT video surveillance systems. Our key idea is to employ a hardware isolation primitive that physically isolates the sensitive hardware and software resources from security compromises. By presenting the countermeasure, we aim to close the loop for the replay threats in IoT video surveillance systems.

To address the software replay attack, we must protect both the DDR memory and the peripheral hardware modules such as the VDMA, so that the malicious software embedded by the attacker cannot compromise these critical system components and issue the replay attack. The basic principle of defense in this case is that trusted software should be authorized to configure and access the PL, while the untrusted software should not gain access.

To realize the aforementioned principle, we develop a hardware isolation framework based on ARM TrustZone [1][38], as shown in Figure 6. The hardware isolation primitive partitions all the hardware and software modules on the SoC into two isolated environments, namely the secure world and the non-secure world. It ensures that the Non-secure World does not have access to the secure world, as the two worlds are physically isolated at the physical bus level, and the access is strictly managed by the ARM processor. For example, in order to access the function or data in the secure world, a normal world application must conduct a secure monitor call (SMC) [1] to switch the context from the normal world to the secure world. In the video surveillance prototype, we place the following system components in the secure world by setting a non-secure (NS) bit to 0: (1) the critical components in the PL part of the SoC, including the motion detection module, the VDMA module, and the HDMI IN/OUT interfaces; and (2) a small secure agent (i.e., a software application) in the PS part of the SoC, which can be employed with access control policies to block illegal accesses to the secure world. The untrusted software (e.g., the malware) in the PS is deployed in the non-secure world by setting the NS bit to 1. In addition, the DDR memory is also split into the two worlds.

Furthermore, Figure 7 shows the underlying hardware architecture of the system realization, which enforces hardware isolation and security. We employ the NS bit at the AMBA bus port, where "NS=1" represents a "non-secure" property, and "NS=0" represents a "secure" property. Then, the AXI Interconnect blocks the access from the normal world under the condition that the NS is set to 0 and the "secure_enable" property is set to 1. In the video surveillance system, we set the NS bit at the AMBA Bus Port to be 0 (i.e., AWPROT[1] for writing and ARPROT[1] for reading) and the "secure_enable" to be 1 at the AXI Interconnect, which combined will block the access from the malicious app to the PL components.

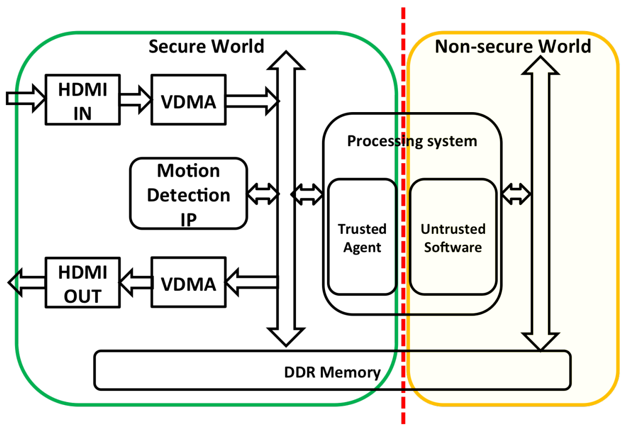


Figure 6: Hardware isolation framework.

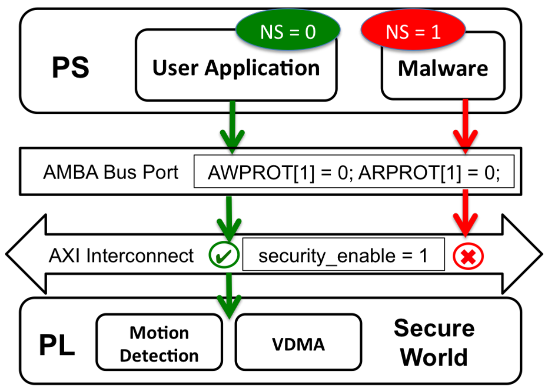


Figure 7: ARM TrustZone configuration to block malicious accesses to the PL.

As a result, in our surveillance video system, even if the attacker embeds a malware, it has to be a separate application in the normal world and viewed by the system as non-secure. The malware does not have the authorization to configure the VDMA module or overwrite the frame buffers that belong to the secure world, and thus it cannot accomplish the "Motion Frame Replacement" step in the software attack flow presented in Figure 5.

5 EXPERIMENTAL RESULTS

In this section, we evaluate both the replay threat model and the hardware isolation-based countermeasure on the video surveillance prototype.

5.1 Replay Attack

We conduct the following two types of evaluations for the performance of the proposed replay attack. (1) the effectiveness of the attack, i.e., whether it can compromise the motion detection module and output the modified video surveillance results; and (2) Difficulty of mitigation, i.e., whether the attack can bypass the existing commonly used security measures.

5.1.1 Effectiveness of the Replay Attack. Figure 8(a) demonstrates the surveillance video system without any attacks. The surveillance video is played on the laptop computer and fed into the motion detection hardware. Then, the motion detection result is delivered to the monitor for display. The white object in the output video represents objects in motion, while the black area represents the background. Because of the random noises in the background, such as waving trees and falling leaves, there may be some random white spots shown in the output video. Once there is a car or a pedestrian passing by, the monitor screen will display a white block with the same shape of the object in motion and at the same position in the video.

Once there is a malware involved in this system, it may store the frames without motion and replay them randomly to mask the motion in the video. As shown in Figure 8(b), there is only random noise shown on the screen despite the presence of motion in the input video. In this case, the tiny difference of the output video from the motion-free video is hardly noticeable, indicating a successfully issued replay attack.

5.1.2 Difficulty of Mitigation. There are several security mechanisms that are typically adopted to protect embedded systems, including encryption, software virtualization, and external hardware security modules. In this subsection, we discuss how the replay attack could bypass these state-of-the-art solutions.

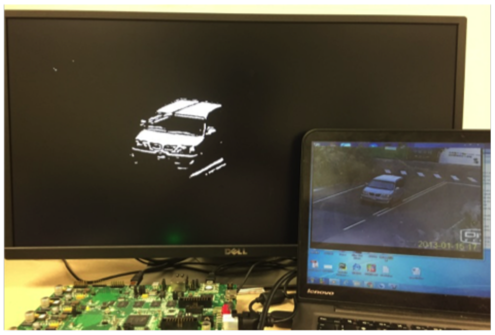
Encryption is the most commonly used data protection method, especially for video content, as the commercially deployed video digital rights management mechanisms today are based on encryption [31][32][34]. With encryption, the data is only exposed to the party that holds the decryption key, which could prevent privacy breaches caused by stealing the sensitive video frames. However, it does not address the security concerns caused by replay attacks, since the attacker can still replace the video frames even if they are encrypted.

Software Virtualization is a popular security scheme that leverages hypervisor and memory management unit (MMU) to manage the access to DDR memory [19][27][18]. However, the replay attack can leverage VDMA to directly access the DDR memory without being blocked by the hypervisor or MMU.

Hardware Security Modules (HSMs), such as trusted platform module (TPM) [25][40][22], leverage a secure hardware chip to store the sensitive data, such as security keys. This mechanism is usually used for authorization and remote attestation. In the replay threat, the malware is an "insider" in the system and can read and write the DDR memory without being blocked by the security mechanisms enforced by the HSMs.

5.2 Hardware Isolation-based Countermeasure

Once the hardware isolation framework is deployed into the surveillance video system, the malware in the normal world cannot access the VDMA module and the memory space located in the secure world. Therefore, the malware will fail to issue the replay attack. However, the hardware isolation framework may introduce additional overhead due to the context switch that is required to execute


(a) Normal Motion Detection

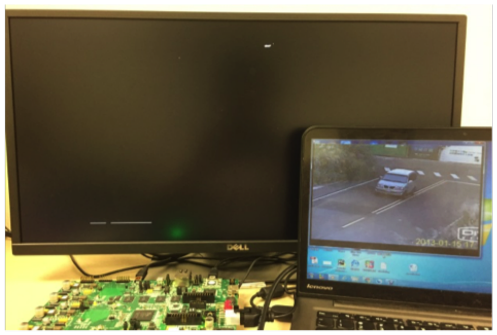

(b) Trojan Attack

Figure 8: Demonstration of motion detection with and without replay attacks.

the application. In this subsection, we evaluate the overhead of the hardware isolation framework from two aspects, namely the timing overhead and the power overhead.

5.2.1 Performance Evaluation. We measure the timing of the surveillance video system with the hardware isolation framework and further compare it with baseline timing results of the original system without protection. In the experiment, we deploy a timer in the software in order to monitor the timing of the motion detection process for each video frame. Furthermore, for the motion detection with hardware isolation, in addition to the motion detection processing time, the timing results also include the round trip world switching time between the secure world and the normal world (i.e., switch from the secure world to the normal world and switch back from the normal world to the secure world).

For both the hardware isolation and the baseline cases, we run the experiments 20 times, and for each execution we collect the single-frame timing results for 50 frames in total. In other words, the dataset includes 1000 single-frame timing results for both cases, as shown in Table 2. We analyze the statistics of the entire dataset by extracting the minimum, first quartile, median, third quartile, and maximum values. By comparing the hardware isolation and baseline results, we observe that the differences, i.e., the hardware isolation overheads, are within 3 μs. Also, the "minimum" value for the hardware isolation case is lower than the no protection case in our experiments, which indicates that the isolation overhead is even smaller than random noises in the timing measurements. In summary, our results indicate that the hardware isolation-based approach does not result in significant timing overhead.

5.2.2 Power Evaluation. We further evaluate the power consumption for the two test cases on the ZC702 board following the power measuring approach in [28]. Figure 9 shows the hardware setup for the power evaluation. We apply the Texas Instruments USB adapter on the ZC702 board and monitor the PS internal power using the TI Fusion Power designer software. In the experiment, we execute the system for both test cases by continuously conducting motion detection for 200 frames. In the mean time, we sample the runtime PS internal power every 250 ms. Figure 10 shows the power results, in which we observe that the power consumption of motion detection ranges from 0.39 W and 0.41W in both cases, and they are almost indistinguishable. It indicates that the hardware isolation mechanism does not introduce noticeable power overhead.

Table 2: Timing evaluation on hardware isolation.

	No Protection (μs)	Hardware Isolation(μs)
Minimum	255,876	255,875
First Quartile	255,886	255,887
Median	255,888	255,890
Third Quartile	255,891	255,893
Maximum	255,900	255,903

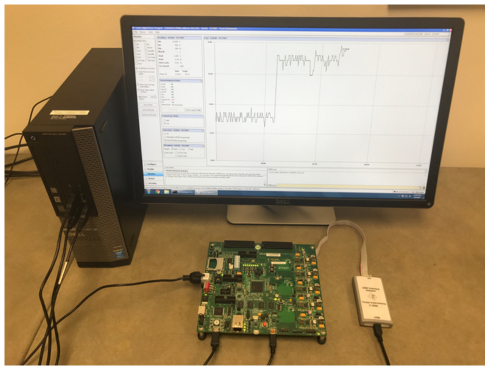


Figure 9: Experimental setup for the power evaluation.

6 RELATED WORK

6.1 Surveillance Video Security

Surveillance video and smart cameras are becoming more and more popular, and the security research on this topic can be divided into two categories. One category aims to enhance the security of the surveillance video itself, which enhances the video coding

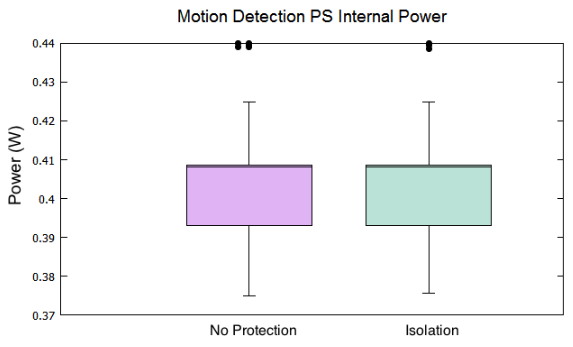


Figure 10: System power evaluation comparing no protection and hardware isolation cases.

mechanism by hiding cryptographic keys in the video content. For example, [24] and [26] introduce an improved approach to protect the video system during the communication by leveraging cryptography and steganography. [6] proposes a hierarchical key generation and distribution system using a multimedia Internet keying protocol. The authors developed different layers of authorization and thus different users could gain access to the content in different layers. The other category studies the video transmission network and the associated threat models. For example, [20] presents a DDoS attack on the video communication network, which causes significant packet losses during video transmission between the client and the server. P2P pollution [13] is another attack method to influence the quality of online P2P video system. Different from the existing studies, our work focuses on the malicious software deployed on smart camera devices.

6.2 Hardware Isolation

Hardware isolation provides the system and upper level applications with a lower level security mechanism by physically isolating the trusted system components from the untrusted ones. It has been supported by several major hardware manufacturers, such as ARM TrustZone [1], Intel SGX [2], and Apple SEP [5]. Such isolation mechanisms have been adopted in a number of security sensitive software applications, such as one-time password token [30], OS kernel [7], software isolation [12], language runtime [21], and cloud computing [23][11][9][39]. To the best of our knowledge, there has been no prior work leveraging hardware isolation to enhance the security of IoT surveillance video systems.

7 DISCUSSIONS ON ADDITIONAL ATTACK SURFACES

In this paper, we have studied the security vulnerabilities of IoT video surveillance systems due to the potential malicious software embedded on the devices. Upon accomplishing this research, we note that the software-based threat model is only one of the several attack surfaces that may challenge sensitive IoT multimedia applications. In this section, we further discuss the additional attack surfaces we identified that may be exposed to compromise the security of the surveillance video use case. Although the detailed discussions and countermeasures for these additional attack surfaces are out of the scope of this paper, we aim to present the rationales behind them at the concept level to motivate further security research along this direction.

7.1 Hardware Trojan Attacks

On a heterogeneous SoC-based smart camera involving both PS and PL, it is often the case that the device manufacturer would outsource the critical system components, such as the motion detection module, to a third party intellectual property (IP) provider. Such a third party module, commonly referred as third party IP core (3PIP) has the potential of significantly reducing the cost of the smart camera device and boosting the deployment of the products.

However, the production and supply chain of the 3PIPs may introduce brand new attack surfaces for the security of the smart camera. For example, the third party manufacturer may not be trustworthy and thus cannot be guaranteed to deliver genuine IP cores that exactly follow the design spec. The potentially modified hardware 3PIPs, if attribute to a malicious intent, are often referred to as hardware Trojans in the hardware security community [33][14]. Although hardware Trojan threats and defense mechanisms in general have been intensively studied [37][10][41][15][8][17][35], we note that the community is still lacking the domain-specific study of hardware Trojan threats and mitigation techniques for the IoT multimedia applications, such as smart video surveillance. The new threat model introduced by hardware Trojans, e.g., a hardware-based replay attack, may pose brand new challenges to the security countermeasures and complicate the entire system design and manufacturing.

7.2 Side Channel-based Attacks

In addition to the security vulnerabilities of IoT surveillance systems studied in this work, we note that the privacy of the surveillance video is also critical as the area under surveillance is often highly private residence or properties. Given the heterogeneous computing architecture on the SoC, it is possible for the adversary to monitor certain side channel information (e.g., timing and power) to infer the specific system operation or even the approximate content of the video. While such a side channel-based attack has been a major thread of research in the software, system, and hardware security communities [34][16][36], its consequences on surveillance video systems have not been fully studied. We further note that the countermeasures for such attacks would be challenging, as the hardware isolation-based approaches we adopt in this work have been known to be ineffective towards side channel attacks by design. Therefore, we foresee that the research along this direction would require additional efforts from the community in order to enhance the security and privacy of the IoT video surveillance system.

8 CONCLUSION

We have developed a motion detection-based IoT surveillance video system and investigated its security vulnerabilities and countermeasures. We first proposed a replay threat model, which hides the objects in motion using pre-recorded non-motion frames. We showed that such a reply attack can be implemented and activated from the software layer of the IoT device, which leaves a huge attack surface to the attackers. Furthermore, we developed a countermeasure against the replay attacks by employing a hardware isolation framework. Our evaluation results on real hardware (i.e., Xilinx Zynq SoC) proves the effectiveness and low overhead of the attack and defense techniques. To the best of our knowledge, our work is the first leveraging hardware isolation to protect IoT surveillance video systems.

9 ACKNOWLEDGEMENTS

We appreciate the constructive reviews provided by the anonymous reviewers. This work was supported in part by the gift donation from Adobe Research and in part by the University of Nebraska Foundation under the Layman Award 1024460.

REFERENCES

[1] ARM Security Technology: Building a Secure System using TrustZone Technology. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html.

[2] Intel Software Guard Extensions. https://software.intel.com/en-us/isa-extensions/intel-sgx.

[3] Nest Cam Spec Sheet. https://content.abt.com/documents/73396/NC2100ES-specs.pdf.

[4] Ring Video Doorbell. https://ring.com/.

[5] 2016. iOS Security Guide. https://www.apple.com/business/docs/iOS_Security_Guide.pdf.

[6] Mamoona Asghar and Mohammad Ghanbari. 2011. Cryptographic keys management for H. 264 scalable coded video security. In *Information Security and Cryptology (ISCISC), 2011 8th International ISC Conference on.* 83–86.

[7] Ahmed M. Azab, Kirk Swidowski, Rohan Bhutkar, Jia Ma, Wenbo Shen, Ruowen Wang, and Peng Ning. 2016. SKEE: A Lightweight Secure Kernel-level Execution Environment for ARM. In *The Network and Distributed System Security Symposium (NDSS).*

[8] Mainak Banga and Michael S. Hsiao. 2010. A region based approach for the identification of hardware Trojans. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST).* 40–47.

[9] Andrew Baumann, Marcus Peinado, and Galen Hunt. 2014. Shielding applications from an untrusted cloud with Haven. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI).* 267–283.

[10] Gedare Bloom, Bhagirath Narahari, and Rahul Simha. 2009. OS support for detecting Trojan circuit attacks. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST).* 100–103.

[11] Stefan Brenner, Colin Wulf, and Rüdiger Kapitza. 2014. Running ZooKeeper coordination services in untrusted clouds. In *USENIX Conference on Hot Topics in System Dependability (HotDep).* 2–2.

[12] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: Minimal hardware extensions for strong software isolation. In *USENIX Security Symposium.*

[13] Prithula Dhungel, Xiaojun Hei, Keith W. Ross, and Nitesh Saxena. 2007. The pollution attack in P2P live video streaming: Measurement results and defenses. In *Workshop on Peer-to-peer streaming and IP-TV.* 323–328.

[14] Jeremy Dubeuf, David Hély, and Ramesh Karri. 2013. Run-time detection of hardware Trojans: The processor protection unit. In *IEEE European Test Symposium (ETS).* 1–6.

[15] Andrew Ferraiuolo, Xuehui Zhang, and Mark Tehranipoor. 2012. Experimental analysis of a ring oscillator network for hardware Trojan detection in a 90nm ASIC. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD).* 37–42.

[16] Ben Gras, Kaveh Razavi, Erik Bosman, Herbert Bos, and Christiano Giuffrida. 2017. ASLR on the line: Practical cache attacks on the MMU. In *Network and Distributed System Security Symposium (NDSS).*

[17] Yier Jin and Yiorgos Makris. 2008. Hardware Trojan detection using path delay fingerprint. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST).* 51–57.

[18] Narjes Jomaa, David Nowak, Gilles Grimaud, and Samuel Hym. 2016. Formal proof of dynamic memory isolation based on MMU. In *International Symposium on Theoretical Aspects of Software Engineering (TASE).* 73–80.

[19] Konstantinos Koukos, Alberto Ros, Erik Hagersten, and Stefanos Kaxiras. 2016. Building heterogeneous unified virtual memories (UVMs) without the overhead. *ACM Transactions on Architecture and Code Optimization (TACO)* 13, 1 (2016), 1.

[20] Chung-Hsin Liu and Chun-Lin Lo. 2009. The analysis of DDoS attack for the video transmission. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human.* 394–399.

[21] Nuno Santos, Himanshu Raj, Stefan Saroiu, and Alec Wolman. 2014. Using ARM TrustZone to build a trusted language runtime for mobile applications. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).* 67–80.

[22] Jared Schmitz, Jason Loew, Jesse Elwell, Dmitry Ponomarev, and Nael Abu-Ghazaleh. 2011. TPM-SIM: a framework for performance evaluation of trusted platform modules. In *Design Automation Conference (DAC).* 236–241.

[23] Felix Schuster, Manuel Costa, Cedric Fournet, Christos Gkantsidis, Marcus Peinado, Gloria Mainar-Ruiz, and Mark Russinovich. 2015. VC3: Trustworthy data analytics in the cloud using SGX. In *IEEE Symposium on Security and Privacy.* 38–54.

[24] Prabira Kumar Sethy, Kamal Pradhan, and Santi Kumari Behera. 2016. A security enhanced approach for video steganography using K-Means clustering and direct mapping. In *International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT).* 618–622.

[25] Jianxiong Shao, Yu Qin, Dengguo Feng, and Weijin Wang. 2015. Formal analysis of enhanced authorization in the TPM 2.0. In *ACM Symposium on Information, Computer and Communications Security (ASIA CCS).* 273–284.

[26] Shikha Sharma and Devendra Somwanshi. 2016. A DWT based attack resistant video steganography. In *International Conference on Information and Communication Technology for Competitive Strategies.* 116.

[27] Matthew Simpson, Bhuvan Middha, and Rajeev Barua. 2005. Segment protection for embedded systems using run-time checks. In *International Conference on Compilers, Architectures and Synthesis for Embedded Systems.* 66–77.

[28] E. Srikanth. 2014. Zynq-7000 AP SoC low power techniques part 2 - Measuring ZC702 power using TI Fusion Power Designer tech tip. http://www.wiki.xilinx.com/Zynq-7000+AP+SoC+Low+Power+Techniques+part+2+-+Measuring+ZC702+Power+using+TI+Fusion+Power+Designer+Tech+Tip.

[29] Chris Stauffer and W Eric L Grimson. 1999. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, Vol. 2. 246–252.

[30] He Sun, Kun Sun, Yuewu Wang, and Jiwu Jing. 2015. TrustOTP: Transforming smartphones into secure one-time password tokens. In *ACM Conference on Computer and Communications Security (CCS).* 976–988.

[31] Viswanathan Swaminathan and Sayaan Mitra. 2012. A partial encryption scheme for AVC video. In *IEEE International Conference on Emerging Signal Processing Applications (ESPA).* 1–4.

[32] Viswanathan Swaminathan and Sheng Wei. 2013. Offline protected video playback on heterogeneous platforms. In *IEEE International Conference on Multimedia and Expo Workshops (ICME).* 1–4.

[33] Mark Tehranipoor and Farinaz Koushanfar. 2010. A survey of hardware Trojan taxonomy and detection. In *IEEE Design & Test of Computers.* 10–25.

[34] Ruoyu Wang, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. 2013. Steal this movie: Automatically bypassing DRM protection in streaming media services. In *USENIX Security Symposium.*

[35] Sheng Wei, Saro Meguerdichian, and Miodrag Potkonjak. 2010. Gate-level characterization: Foundations and hardware security applications. In *Design Automation Conference (DAC).* 222–227.

[36] Sheng Wei, James B. Wendt, Ani Nahapetian, and Miodrag Potkonjak. 2014. Reverse engineering and prevention techniques for physical unclonable functions using side channels. In *Design Automation Conference (DAC).* 1–6.

[37] Francis Wolff, Chris Papachristou, Swarup Bhunia, and Rajat S. Chakraborty. 2008. Towards Trojan-free Trusted ICs: Problem analysis and detection scheme. In *Design, Automation and Test in Europe (DATE).* 1362–1365.

[38] Xilinx Inc. 2014. Programming ARM TrustZone Architecture on the Xilinx Zynq-7000 All Programmable SoC. In *UG1019 (v1.0).*

[39] Yan Zhai, Lichao Yin, Jeffrey Chase, Thomas Ristenpart, and Michael Swift. 2016. CQSTR: Securing cross-tenant applications with cloud containers. In *ACM Symposium on Cloud Computing (SoCC).* 223–236.

[40] Dawei Zhang, Zhen Han, and Guangwen Yan. 2010. A portable TPM based on USB key. In *ACM conference on Computer and Communications Security (CCS).* 750–752.

[41] Xuehui Zhang, Andrew Ferraiuolo, and Mohammad Tehranipoor. 2013. Detection of Trojans using a combined ring oscillator network and off-chip transient power analysis. *ACM Journal on Emerging Technologies in Computing Systems* 9, 3, Article 25 (2013), 25:1–25:20 pages.