Analyzing the Security and Privacy of Cloud-based Video Surveillance Systems

Johannes Obermaier
Fraunhofer Institute AISEC
Parkring 4
Garching b. München, Germany
johannes.obermaier@aisec.fraunhofer.de

Martin Hutle
Fraunhofer Institute AISEC
Parkring 4
Garching b. München, Germany
martin.hutle@aisec.fraunhofer.de

ABSTRACT

In the area of the Internet of Things, cloud-based camera surveillance systems are ubiquitously available for industrial and private environments. However, the sensitive nature of the surveillance use case imposes high requirements on privacy/confidentiality, authenticity, and availability of such systems. In this work, we investigate how currently available mass-market camera systems comply with these requirements. Considering two attacker models, we test the cameras for weaknesses and analyze for their implications. We reverse-engineered the security implementation and discovered several vulnerabilities in every tested system. These weaknesses impair the users' privacy and, as a consequence, may also damage the camera system manufacturer's reputation. We demonstrate how an attacker can exploit these vulnerabilities to blackmail users and companies by denial-of-service attacks, injecting forged video streams, and by eavesdropping private video data — even without physical access to the device. Our analysis shows that current systems lack in practice the necessary care when implementing security for IoT devices.

Keywords

Security Analysis; Surveillance Systems; Internet of Things; Home Automation; Privacy; Embedded Security; Pentesting

1. INTRODUCTION

The emerging field of *Internet of Things* has been populated by numerous devices in the last few years. The decreasing costs and growing availability of embedded devices enabled Smart Home and Home Automation systems. One of the typical applications are video surveillance systems for building security. These devices often feature a connection to a cloud server or other relays, so that the user has access from all over the world. But these new possibilities are accompanied by new threats [5], for the user as well as for the offering company.

IoTPTS'16, May 30-June 30 2016, Xi'an, China

DOI: http://dx.doi.org/10.1145/2899007.2899008

Cameras are often employed in spaces, where privacy must be guaranteed. As shown by recent surveys, users are concerned about preserving their privacy [2] and fear unauthorized manipulation [3], especially in cloud-based services. Therefore, the expectations on manufacturers with respect to data privacy, integrity, and system availability, are high. For example, it must be ensured that users are not eavesdropped, their video stream is not manipulated, and system operation remains uninterrupted, as also mentioned in [6]. Conflicting with these requirements would not only be disadvantageous for the user but also leads to problems for the manufacturing companies, by blackmailing or loss of reputation.

Very little information is supplied by manufacturers on how a system communicates with its cloud server and whether the connection is well-secured. The security concepts are usually not standardized, are mostly proprietary, and therefore not disclosed, hereby rendering public reviews impossible. Furthermore, it is often not even ensured that cryptographic algorithms are used at all, not to mention a secure and technically correct implementation.

The key elements for security in surveillance systems received a lot of attention, but have been seldom verified for real applications. In 2009, Vagts and Beyerer state several challenges, for example data privacy, and conclude, that "the issues must be addressed right from the beginning" [7]. He emphasizes "security by design" and "privacy by design". An overview of several implementations of data-centric security approaches is given in [8]. The concepts are compared regarding their ability to provide integrity, authenticity, data freshness, confidentiality, and related key elements. In 2015, a prototype for secure embedded visual sensing was published [9]. The work focuses on end-users and especially takes cloud storage into account. Beside that there are several proposed methods and concepts, their realization in currently available end-user solutions is unclear and needs further investigation. For embedded devices in general, automated large-scale firmware analysis has been conducted in [1]. The researchers were able to extract password hashes and RSA keys, for example. But these tests only cover specific patterns. In contrast to this, we perform a deep evaluation of the camera systems and investigate the implications of identified weaknesses in specific devices.

In our work we analyzed the quality of the employed security concepts in four cloud-based video surveillance systems. A focus was put on the design and implementation of communication protocols. Our approach uses disassembling of the camera firmware and traffic analysis. The implemen-

tations were assessed regarding their ability to provide *privacy*, *integrity*, and *availability*. Related subjects, like cloud server penetration tests and hardware attacks using physical access to the camera, are not considered in our work. They are either not legally feasible or they are not of a significant relevance in real world applications. Two network-based attackers are defined, which threaten the end-user and the manufacturer.

In all four systems we found major security issues, namely security design and implementation weaknesses. Some systems allow an attacker to perform very simple denial-of-service attacks. In others, injecting forged video streams is possible and attackers can even eavesdrop the users' video stream.

2. MATERIAL AND METHODS

This section describes the selected camera systems, the technical approach for analyzing the systems, and defines two attacker models. Due to the severity of the discovered issues, the names of manufacturers and systems are withheld. All results were previously shared to their full extent with the manufacturers.

2.1 Selected video surveillance cameras

For our study, we selected standalone solutions, where the user only has to provide power and network connectivity to a *single* device. Complex systems for large scale surveillance were not considered. We chose only cameras that communicate with a cloud server or a gateway service, and offer the possibility to access the video stream using the Internet. The chosen systems are a non-representative sample that cover low-cost solutions as well as high-priced devices.

Camera A is a medium-priced camera system with extended features. It can store video streams in the cloud for later viewing. Additionally, the camera supports motion detection.

Camera B is a high-priced system, which is technically related to Camera A. The features of both systems are the same. We could verify, that both cameras are based on the same hardware, firmware, and a similar cloud software. The camera is distributed by another company, which chose to include some of their own modifications.

Camera C is a camera of the lower price segment. There is no cloud storage but a gateway solution, which helps establishing the connection between the user and the camera. The user can view the video stream using an application or a web browser.

Camera D is also a camera of the lower price segment. It also does not support cloud storage for recording. The video stream can be viewed over the Internet using a relay technique, which uses the cloud servers.

All cameras are running an embedded Linux operating system and can be connected to the Internet using wired Ethernet or a wireless network.

2.2 Attacker model

Two attacker models were defined for the security analysis. The first one describes the *local attacker* and the second one constitutes the *remote attacker*.

The **local attacker** is characterized by the access to the local network that serves the Internet connection to the camera device. The attacker is able to exchange data with the network like a usual node. Although this attacker might be inside the building, physical access to the camera is not assumed. A malevolent employee or guest represents such an attacker.

The **remote attacker** does not have access to the local network and is only equipped with Internet connection. The attacker is therefore able to reach the cloud servers, but cannot directly communicate with the cameras. We extend the typical remote attacker by the ability to observe encrypted wireless network traffic. While sometimes this ability is attributed to the local attacker, we consider this in the remote attacker model, since here no access to the building is necessary. This extension enables some special attacks, which can be executed in vicinity of camera systems, but without requiring physical access to the device, building or internal network.

Both attackers are considered to be technically experienced and have access to a simple electronics lab. It is assumed, that they can purchase a camera as a usual customer.

2.3 Technical approach

For both attacker models the same technical approach is used. The analysis first aims at reverse-engineering the systems' security concepts and at understanding their way of implementation. Two main methods are employed therefore: Traffic analysis and firmware disassembly.

Traffic is analyzed by an interposed computer system, which acts as a router. For packet recording and forwarding, the tools "Wireshark" and "iptables" are employed. But traffic analysis can only extract basic properties, whereas firmware disassembly gives a much deeper insight. Therefore the firmware of each device was extracted, in most cases by unsoldering and reading out the flash memory IC. The binaries were analyzed using the reverse-engineering framework "radare2".

In a second step, the gathered information was used for a thorough security analysis and investigation of implications of the findings.

3. SECURITY ANALYSIS OF THE DEVICES

In this section, the results of the security analysis are presented. Due to the large number of discovered weaknesses, only the key issues are explained in detail. At first, the connection to the cloud is described in general. Next, the security of the communication protocols and their encryption algorithms are analyzed and assessed. Finally, the method of camera authentication is reviewed and rated regarding its trustworthiness.

3.1 Analysis of communication encryption

The first step of the analysis comprises the identification of the connection types and the mechanisms for securing the different connections. Herefore, the traffic between the camera and the cloud is monitored and analyzed with Wireshark. Basically, all four cameras employ the following three type of connections. Table 1 summarizes our findings.

The *initial* connection is used for a first communication with the cloud servers. In addition it is used for camera-is-online and motion-detection events. All cameras use some kind of encryption here: Camera A and Camera B establish a connection using TLS 1.2, Camera D uses SSL 3.0. In contrast to this, Camera C relies on a non-standardized method, which uses a proprietary encryption algorithm and protocol, based on UDP.

Connection	Camera A	Camera B	Camera C	Camera D
initial	TLS 1.2	TLS 1.2	proprietary	SSL 3.0
configuration	none	proprietary		
data	none		none	none

Table 1: Overview of secured protocols used for the cameras

The *configuration* connection, which is used to set up video stream parameters and can trigger its transmission, is treated differently: While Camera C and Camera D use the previously established encrypted channel, Camera A and Camera B establish a new connection. Although Camera A and Camera B are based on the same system, their behavior differs: Camera A utilizes an unencrypted human-readable proprietary protocol. Camera B employs an encrypted proprietary protocol instead.

To transmit the actual *data*, i.e., the video stream, most systems fall back to unsecured connections. Only Camera B reuses its secured proprietary configuration connection.

The next step is a deeper analysis of all secured protocols. Here severe issues were found for the TLS connection of Camera B, the proprietary protocol of Camera B, and the proprietary protocol of Camera C.

3.1.1 Weak certificate validation in the TLS connection of Camera B

All cameras have the cloud server's public RSA certificate stored in the flash memory. When the cloud is contacted, the server certificate is validated by the camera during the handshake. In the case of a certificate error, the connection is closed and later retried.

This behavior is correctly implemented in the TLS resp. SSL implementations of Camera A and Camera D. But the reaction of Camera B was surprising. After encountering a certificate error, the system fell back to an unsecured connection using HTTP. The camera log files revealed that this is the intended behavior: "SSL failed for HA proxy event. Fallback to HTTP."

This implementation counteracts the idea of validating the communication partner and encrypting traffic. Typical MITM attacks become viable. The *local attacker* could interpose himself/herself between the camera and the cloud server. One possible action is controlling the device and changing its settings.

3.1.2 The proprietary encryption of Camera B

We now analyze the proprietary communication protocol Camera B uses the in the configuration and data connections. Each communication is initiated with the transmission of the string `HELLO` and the camera's MAC address, to which the server responds with `CHALLENGE` and a 128-bit nonce. From that point on, the communication is encrypted.

The data does not show any structure nor any other anomaly. An entropy test showed a value of approximately 7.99 bits per byte, which is a hint at a strong encryption algorithm. Traffic analysis did not expose more details and thus the firmware was disassembled for a deeper insight.

The protocol is built up using the cryptographic primitives SHA-1 and AES128. Their implementation is highly optimized and therefore most likely library code. The speed-optimized implementation of the AES128 features an almost branch-less implementation and makes use of T-tables.

The AES session key derivation is implemented as follows. When a connection is being established, the server sends the aforementioned 128-bit nonce N. The nonce and the hard-coded 128-bit pre-shared key K_p are concatenated to the 256-bit intermediate key K_i. Please note, that K_p is a constant in the application binary and shared by *every* device. By applying SHA-1 on the intermediate key K_i, the 160-bit hash K_h is generated. The hash K_h is truncated to 128 bits by discarding four bytes at its end. This creates the the 128-bit AES session key K_a.

The encryption and decryption is based on AES128 in cipher feedback mode (CFB). It uses the AES session key K_a as long as the connection persists. Surprisingly, the initialization vector of the CFB mode is set to zero in the beginning. The CFB state is re-initialized with zero each time the direction of communication changes or whenever a new dataset is received. Thus the same initialization vector is used repeatedly.

Altogether, two major weaknesses were discovered in the protocol design: The reuse of the zero initialization vector and the deployment of the same pre-shared key on all devices. The first vulnerability weakens the encryption and an attacker may be able to reconstruct parts of the CFB keystream. The second weakness allows to generalize the vulnerability to all cameras of this type. This issue effectively renders the encryption useless, since an attacker can encrypt and decrypt arbitrary packets of any system with the knowledge of the pre-shared key.

3.1.3 The proprietary encryption of Camera C

The manufacturer of Camera C developed a proprietary encryption and protocol.[1] Naturally, there is no information on this protocol publicly accessible and the protocol does not follow any known standards. The data does not contain any human-readable contents nor is there a typical header, handshake message, or certificate exchange. A visual inspection quickly reveals that the packet data contains patterns and repeating byte sequences.

A statistical analysis of packet data showed that the byte values are not evenly distributed. Each packet has a significant accumulation of several specific values, while others are underrepresented. A histogram of the byte values of a single packet are depicted in Fig. 1. The distribution is similar for all packets, but the horizontal position of the peaks depends on the selected packet. The entropy of typical data was determined as being between 5 and 6 bits per byte. This is far from well-encrypted data that approaches almost 8 bits per byte entropy. The findings lead to the conclusion, that there exists a security flaw in the protocol and cryptographic algorithm design.

The encryption algorithm was reverse-engineered by disassembling of the camera's firmware. The corresponding binary contains the symbols `lrotate`, `rrotate`, and `choose_random_shiftnum`, which are located in vicinity of communication code. These functions constitute the core of

[1]This was later also confirmed by a support employee.

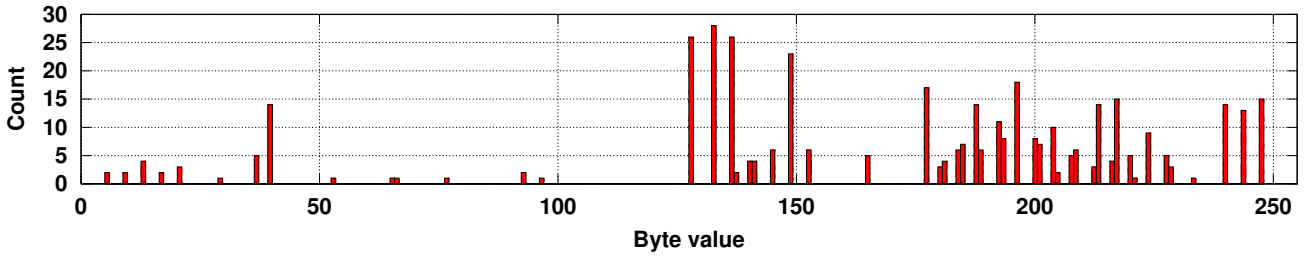

Figure 1: The statistical distribution of byte values of one encrypted packet (404 Bytes). The imbalance hints at a major security flaw.

the "encryption": The bits in each byte are circularly right-rotated by a random number of positions between one and seven. This number, plus the constant `0x3C` (ASCII character `"<"`), is stored in the first byte of each packet.

By applying the inverse of the extracted algorithm, all packets can easily be decoded. A simple XML dataset is revealed, which is also human-readable and contains all camera control data. The decryption requires no key, the knowledge of the algorithm suffices. Thus this encryption is more a weak obfuscation, than a secure algorithm.

With this knowledge, an attacker is now able to decode the communication and can also communicate with the cloud servers. This leads to a severe threat, as it will be shown in Section 4.3.2.

3.2 Authentication and access control

3.2.1 Weak factory interface password at Camera A and Camera B

All devices can be reached over the local network at port 80 (HTTP). Camera C and Camera D offer a configuration interface to the user. Both of them are protected against unauthorized access by requiring login credentials.

Camera A and Camera B also offer an interface, but the user is not supposed to access it, the credentials are not disclosed to the user. The cameras can only be configured using the cloud service. There is an *individual* password for each device, which is stored in the camera's flash memory. The password has a length of 36 characters, contains characters of mixed case and also numbers. Theoretically, this would be a sufficiently strong password. Despite the password is individual for each device and seems to be long enough, it has shown to be very weak. The password was identified as Base64 encoded string. When the inverse function was applied, the method of password generation became evident. The password is the Base64 encoded concatenation of two (unpublished) constant strings with the reversed MAC address. For the device with the MAC address 01:23:45:67:89:AB, the password is `base64encode([...]BA9876543210[...])`. With this knowledge, an attacker can compute the password for *any* camera, given its MAC address. The local attacker gains unlimited access to the camera's factory interface.

3.2.2 Device and stream authentication of Camera A and Camera B

Both cameras, Camera A and Camera B, are based on the same concept to authenticate the camera to the cloud server. Despite Camera B encrypts all data, it can be regarded as being as secure as Camera A, since the encryption of its proprietary protocol is practically ineffective. Thus it is appropriate, to apply the following results to both systems.

After the configuration connection to the cloud server has been established, the camera has to identify itself. The server identifies the camera by the means of its unique MAC address, which is the only identifying information that is transmitted by the camera. Thus there is no authentication, but only a weak identification.

Remember that both cameras use for the initial connection into the cloud a TLS 1.2 connection that does not include client-authentication. The payload is a usual HTTP GET-request to the cloud server. The devices access the scripts `httpevent.php` and `httpmotionevent.php`. The first one is polled regularly and generates a camera-is-online event. When the second one is accessed, it causes a motion-event notification. The cameras identify themselves by appending their own MAC address as a GET-request parameter.

Altogether, no authentication, but only a identification is employed in the system. This is a major security design flaw, which enables a range of attack vectors for the remote attacker. Since there are no means for a secure authentication, the impersonation of cameras becomes a major threat.

3.2.3 Device and stream authentication of Camera C

Camera C uses a similar authentication scheme that is also based on the camera's MAC address. This procedure takes place inside the encrypted connection. As shown in Section 3.1.3, the self-made encryption algorithm is futile and can easily be deciphered. Thus the camera authentication data is clearly readable for analyzing.

After the connection to the cloud server has been established by the camera, it sends an XML-encoded dataset. The camera identifies itself by including the field `id`, whose value is the MAC address, into the dataset. Other data, like serial numbers, are also transmitted, but obviously not used for authentication, the MAC address suffices. Thus the same threats emerge, where an attacker can impersonate the camera.

3.3 Server-side issues

3.3.1 The geolocation service pin code of Camera D

Camera D queries the cloud server for geolocation information upon startup using HTTP. The device determines its own location in order to adjust the timezone setting, for example. The cloud server processes the query depending on the client's IP address. Such services are offered by com-

Attack	Camera A	Camera B	Camera C	Camera D
Camera impersonation	R	R	R	–
Server impersonation	L	L	L	–
Full manipulation of the camera	L	L	–	–
Stream eavesdropping	L	L	R	(L)
Attack on server	–	–	–	R

Table 2: Overview of possible attacks by a local attacker (L) and a remote & local attacker (R)

panies, which sometimes have a pay-per-use policy, so each request causes a small amount of costs — or, if there are no costs, at least generates server load. Regardless, the protection of the service is crucial.

In order to defend the cloud service against misuse by unauthorized clients, a pin code is employed. When the device sends its request, it has to include its own MAC address and the corresponding pin code. Only if the MAC address is authorized and the pin code matches *for this* MAC address, a reply will be delivered.

Analyses discovered, that the pin code is generated on-demand on the camera system. The algorithm was extracted from an executable file. The pin code is the MD5 hash of an (undisclosed) 64 byte long string that consists of constant characters and parts of the MAC address. E.g., pincode = `md5([...]01:23:45[...]01:23:45:67:89:ab[...])`

Using this knowledge, the remote attacker is able to generate valid pairs of *arbitrary* MAC addresses and pin codes. Abuse of the cloud service becomes viable, high system load and costs can easily be caused.

4. ATTACKS

We show in this section, how a local and remote attacker can exploit the previously described security design and implementation weaknesses. Due to the high number of potential attacks, only the most severe ones are described. The mentioned attacks were successfully verified in practice.

4.1 Summary of found issues

A summary of vulnerabilities is given in Table 2. For each camera, at least one remote and several local vulnerabilities were discovered.

4.2 Local attacker

The local attacker creates a major threat for Camera A, Camera B, and Camera C, due to their insufficiently secured factory interface. Furthermore, the cloud server identity is not always verified, which creates another vulnerability. All cameras, including Camera D, have no or only weak stream encryption, which enables eavesdropping.

4.2.1 Factory interface abuse

Factory interface abuse is feasible for Camera A and Camera B. After computing the password, the attacker gains full access to the camera's factory interface. The attacker can then view and record the current video and audio stream, despite the fact that she is officially not authorized to do so. Additionally, continuous surveillance is achieved by abusing a feature, which supports the periodic transmission of an image to any external SMB server. These attacks can easily impair another person's privacy.

The camera can also be manipulated or even destroyed using further actions. If invalid configuration data is set, the system becomes unusable or will be damaged. In some cases, a manual reprogramming of the non-volatile memory was necessary to fix the issue — a step which commonly is impossible for the user. This very simple attack causes a very effective denial-of-service effect, which can practically destroy the system and thus impede system availability.

The attacker is also able to gain full access to the underlying operating system by exploiting another weakness of the factory interface. Input sanitizing is not performed for configuration data and thus command injection is possible. The attacker may enable a remote root shell and can then manipulate any data, including the camera's firmware. Any attack from device destruction up to the infringement of privacy becomes feasible.

These issues are especially severe, since the user is neither able to detect the attacker nor able to protect the camera. The existence of such an interface is not documented and it is therefore unknown to the user.

4.2.2 Cloud server impersonation

Cloud server impersonation is feasible for Camera A, Camera B, and Camera C. Slightly different approaches must be applied for each system, but the outcome is similar. The attacker obtains control of most of the camera's function and can perform manipulations. This can be used for denial-of-service attacks and also for privacy infringement.

4.3 Remote attacker

All four cameras and the cloud infrastructure are threatened by the remote attacker. While there is only a minor weakness in Camera D, the other three cameras show major issues. In the case of Camera A, Camera B, and Camera C, the impersonation of cameras is possible. This enables an attacker to inject forged video streams. For Camera C, even unauthorized access to the camera and its stream is possible, using a more complex procedure.

For all these attacks, a MAC address is needed. It is acquired using the following two methods, depending on the intention of the attacker. If a specific camera shall be attacked, it is sufficient to receive a single, maybe encrypted, packet from the camera's wireless LAN interface. Since the MAC address is included in the unencrypted packet header, it can be extracted immediately. In the other case, that random or many cameras shall be attacked, MAC addresses can simply be guessed. Each manufacturer has only a limited range and they are usually used incrementally. Based on a known MAC address, the attacker can enumerate a large number of further addresses, for which it is likely, that they exist.

4.3.1 Camera impersonation (Camera A, Camera B)

The cameras Camera A and Camera B can be impersonated. There is no secure authentication, but only a weak identification of the camera by means of its MAC address.

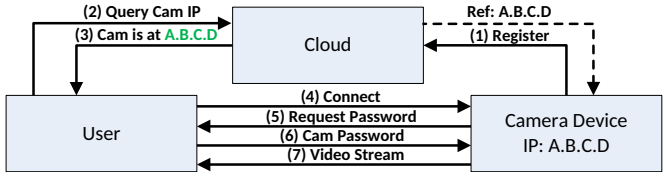

Figure 2: The mode of operation in the undisturbed system.

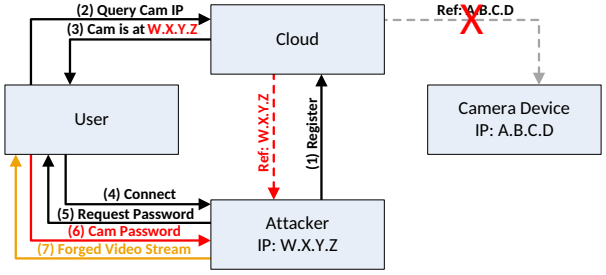

Figure 3: The attacker tries to get the user's camera password by impersonating the camera. Afterwards, the attacker is also able to send a forged video stream to the user.

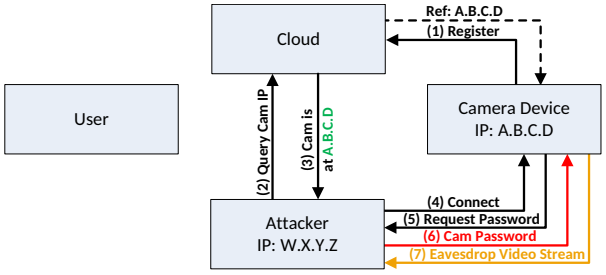

Figure 4: The attacker logs into the camera using the previously acquired password. He can then eavesdrop the user's video stream or change settings.

The attacker can trigger a motion-detection event, by sending a TLS encrypted HTTP GET-request to the cloud server. The request must access the httpmotionevent.php script and has to contain the camera's MAC address. For example: `https://[...]/httpmotionevent.php?[...]&mac=0123456789AB`

The user will be alarmed about a detected motion. Since the attacker can trigger such an event easily, users can not rely on the authenticity of the reported incident.

A denial-of-service attack is also possible, using the control connection. In the case of Camera B, the data must be encrypted before sending, but this can easily be performed with the knowledge of the implementation and pre-shared key. Apart from that, the attack is similar for Camera A and Camera B. To start the attack, an attacker establishes the control connection to the cloud server and identify herself as camera. This action will immediately cease the cloud server's connection to the corresponding real camera. The server only preserves the latest connection. If the attack is repeated every 40 to 50 seconds, the camera remains disconnected continuously. Thus the user can no longer access the camera. Since the required communication packets are very small and sum up to less than one kilobyte, the attack is scalable. The attacker can easily switch off numerous cameras with a single Internet connection. Thus this is a typical denial-of-service attack, which can range from a single user up to hundreds or even thousands of devices.

The attacker is also able to inject a forged video stream. Instead of only blocking the camera, the attacker behaves like a real camera at the configuration connection. When the cloud server requests a stream, the attacker can reply with an arbitrary video stream, which will then be displayed to the user and/or recorded in the cloud. The attacker may choose any video, ranging from a black still image up to an intimidating and disturbing video stream. This attack targets the availability but even more the authenticity of the image data. It is a major threat also for the manufacturing company, because their reputation is impaired by such attacks. Due to the high visibility and large effect, this attack may be very promising for blackmailing a company.

4.3.2 *Camera impersonation and stream eavesdropping (Camera C)*

Camera C is affected by the same issues as Camera A and Camera B. Since there is no secure authentication, denial-of-service attacks and the injection of forged video streams are possible. Availability and authenticity of camera data cannot be ensured. Thus the system suffers from similar threats, which may harm the user and especially the manufacturing company.

There exists a related method for Camera C that allows an attacker to eavesdrop the original video stream. Each time the user wants to watch the current video stream, a connection with the camera is established. The camera requests a password from the user. It is then transmitted back to the camera for validation, as depicted in Figure 2.

The attacker can exploit this behavior in order to gain knowledge of the camera's password. Therefore, the camera is impersonated as previously described and shown in Figure 3. When the user wants to view the video stream, the connection is not established to the real camera, but instead to the attacker. The attacker is now able to request the password from the unaware user. The password is transmitted back to the attacker, who gains knowledge of the camera's password.

Then the attacker ceases the impersonation and the real camera takes its place again. The attacker is able to login into the real camera and eavesdrop the video stream, using the official software and the acquired password. These steps are shown in Figure 4. This impedes the confidentiality of the user's camera and harms the user's privacy. Additionally, the manufacturer is threatened even more, since such incidents can have a very large impact on its reputation.

5. CONCLUSION

This paper presented a security analysis of four video surveillance systems. While being maybe considered as simple consumer devices, they have a significant impact on privacy and e.g. building security. Furthermore, we have strong indications, that some of the manufacturers use the vulnerable software components also in other IoT devices beside camera systems.

Our analysis showed significant flaws in the design and implementation of the devices under test. Interestingly, most of the vulnerabilities stem from very common mistakes in security design.

Fallback to unsecure function Being able to downgrade a function or protocol to use an unsecure method is a very common pattern in recent attacks (cf. eg. the FREAK attack [4]). In case of Camera B, a certificate validation error (which should have dropped the connection) led to a fallback to an unsecured connection.

Proprietary security protocols Writing your own crypto is usually a very bad idea. It is already hard for well-established cryptographic primitives to avoid systematic and implementation flaws. Any self-made implementation will very likely fail. In case of the proprietary encryption of Camera C, the method does not follow the Kerckhoff's principle, and the algorithm (which is in general weak) could be extracted by reverse engineering. On the other hand, inappropriate handling of the initialization vector, as it happened in the proprietary encryption of Camera B, is a very common implementation fault of otherwise secure primitives.

Weak passwords It is well-known that default passwords or master keys are a significant threat in general, as such passwords can be revealed by analysis of a single device and then used for large-scale remote attacks. In case of Camera B, there is a static key used in the encryption. Furthermore, in this camera as well as in Camera A, the factory interface password is generated from a public information, namely the MAC address of the camera.

No secure authentication All tested cameras have problems with authentication. In case of the cameras A, B, and C, the identity of a camera is verified by only checking the MAC. Similar, Camera D uses in addition a pin code, which, however, can be generated from the MAC in by using a static function.

Lack of security engineering The overall problem is the missing/insufficient consideration of security during the development of the product. A thorough security analysis according to well-established standards would have identified probably revealed many of the problems that were found in this evaluation. In such an analysis, in case of IoT devices, not only remote attacks but also local and physical attacks need to be considered.

The attacks that were possible due to these vulnerabilities (presented in Section 4) showed the criticality of the problems and the need for a secure engineering process for such devices. Although only four cameras were tested, our findings indicate that current IoT devices often do not follow these basic techniques for state-of-the-art security.

6. REFERENCES

[1] A. Costin, J. Zaddach, A. Francillon, D. Balzarotti, and S. Antipolis. A large-scale analysis of the security of embedded firmwares. In *USENIX Security Symposium*, 2014.

[2] Deloitte & Technische Universität München. Ready for Takeoff? Smart Home aus Konsumentensicht. *http://www.connected-living.org/content/4-information/5-downloads/4-studien/5-ready-for-takeoff/deloitte-smart-home-consumer-survey-20150701.pdf*, July 2015.

[3] GfK. Smart home beats wearables for impact on lives, say consumers. *http://www.gfk.com/fileadmin/user_upload/dyna_content_import/2015-11-24_press_releases/data/Documents/Press-Releases/2015/2015-11-11_smart-home_press-release_global.pdf*, November 2015.

[4] M. Green. Attack of the week: FREAK (or 'factoring the NSA for fun and profit'). http://blog.cryptographyengineering.com/2015/03/attack-of-week-freak-or-factoring-nsa.html, Mar. 2015.

[5] P. Kocher, R. Lee, G. McGraw, and A. Raghunathan. Security as a new dimension in embedded system design. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, pages 753–760, New York, NY, USA, 2004. ACM. Moderator-Ravi, Srivaths.

[6] N. Serpanos and A. Papalambrou. Security and privacy in distributed smart cameras. *Proceedings of the IEEE*, 96(10):1678–1687, 2008.

[7] H. Vagts and J. Beyerer. Security and privacy challenges in modern surveillance systems. In *Proceedings of the Future Security Research Conference*, pages 94–116, 2009.

[8] T. Winkler and B. Rinner. Security and privacy protection in visual sensor networks: A survey. *ACM Computing Surveys (CSUR)*, 47(1):2, 2014.

[9] T. Winkler and B. Rinner. Secure embedded visual sensing in end-user applications with trusteye. m4. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pages 1–6. IEEE, 2015.