dokspot – Securely Linking Healthcare Products with Online Instructions

Kevin Lapagna, Moritz Zollinger, Marc Rennhard
School of Engineering
Zurich University of Applied Sciences
Winterthur, Switzerland
Email: lkev,zolg,rema@zhaw.ch

Hans Strobel, Cyrille Derché
dokspot GmbH
Zurich, Switzerland
Email: hans.strobel,cyrille.derche@dokspot.com

***Abstract*—Printed instructions for products get replaced more and more by digital versions that are made available over the Internet. In safety-sensitive fields, such as healthcare products, availability and integrity of these instructions is of highest importance. However, providing and managing instructions online opens the door to a wide range of potential attacks, which may negatively affect availability and integrity. In this paper, dokspot is presented, which is an Internet-based service that aims at solving this problem by securely linking healthcare products with online instructions. The key to achieve this is a sophisticated security architecture and the focus of this paper is on the core components of this architecture. This includes a secure workflow to manage online instructions, which prevents, e.g., attacks by malicious insiders. Also, the traditionally monolithic web application architecture was split into role-based microservices, which provides protection even if parts of the system are compromised. Furthermore, digital signatures are utilized to continuously safeguard the lifecycle of online instructions to guarantee their genuineness and integrity. And finally, a passwordless signature scheme is introduced to hide inconvenient extra steps from the users while still maintaining security. Overall, this security architecture makes dokspot highly resistant to a wide range of attacks.**

Keywords–Web Application Security; Microservices; Digital Signatures; Passwordless Signatures; Healthcare Product Instructions; Online Document Management System.

I. INTRODUCTION

Today, products should be designed in a way that allows intuitive and safe use without reading the instructions. However, with increasing risks of using a product, relying on intuition and using a "trial and error" approach is unacceptable and becomes a potential safety risk. An illustrative example is the operation of a passenger airplane, where pilots have to read and tick off instructions every time before operating the aircraft. Everybody would agree that in this scenario, doing it in this way and by using the right instructions is an important safety factor.

Another product category that requires detailed knowledge of the instructions are healthcare products, which – in the context of this paper – includes any substance, product, or system used for therapeutic or diagnostic purposes on the human body (or animals). Healthcare professionals (doctors, nursing staff, operators of medical machines, etc.) must be aware of all details involving the use of a healthcare product prior to its application on a patient. To achieve this, the healthcare professional must have guaranteed and simple access to the right instructions in the right language at the right time.

To comply with this, manufacturers of healthcare products predominantly ship printed instructions in multiple languages together with their products. This has various disadvantages, including that the related costs – financial, operational, and environmental – are substantial, that most of the included instruction languages remain unused, and that the required instructions can often not be found when needed (e.g., because they were misplaced or thrown away). Also, it is sometimes necessary to break a product seal to get to the printed instructions, which implies the product can often not be returned to the manufacturer if it turns out the product is not suited for the planned application.

Some of these limitations can be remedied by providing the instructions online. This is typically done with a web portal where healthcare professionals can search for and download instructions of specific products. While this sounds to be a good solution, it has its limitations in practice. One limitation is that every manufacturer uses its own portal, which means the healthcare professional not only has to find the right portal, but also has to be able to cope with different user interfaces. Another limitation is that once the right portal has been found, it may be difficult to find the correct instruction for a specific product. A third limitation is that such a portal may be an attractive attack point, e.g., for competitors or for outside attackers, as tampering with the provided instructions in malicious ways may have a devastating effect on patient safety.

To overcome these limitations, dokspot was developed. dokspot is a novel Internet-based service that aims at transforming the way companies handle instructions by providing an innovative paperless and trustworthy solution. Figure 1 depicts the basic functionality of dokspot.

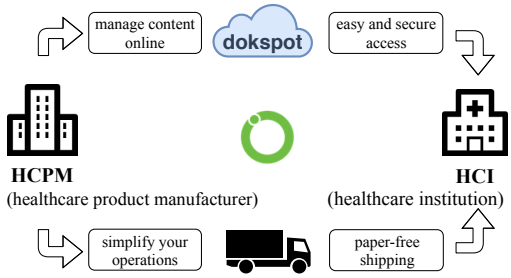

Figure 1. Basic Functionality of dokspot

Figure 1 shows that manufacturers of healthcare products

(left side, these are the customers of the dokspot service) upload their instructions to dokspot (top). Users of the products, in this example healthcare professionals that are typically working at healthcare institutions (right side, these are the customers of the manufacturer of the healthcare products), can easily access this information using the dokspot service when they need to do so. Of course, the physical products must still be shipped to the healthcare institutions (bottom), but they do not include instructions on paper. Among other advantages, this simplifies operations for the manufacturers of healthcare products and reduces hospital waste.

To link a physical product to the online instructions, a Uniform Resource Locator (URL) pointing to the dokspot service is used. This URL is defined by the manufacturer of the healthcare product and communicated to the healthcare professional (the user of the device) together with the product, e.g., on the product label. Figure 2 shows an example of such a label, which includes the URL on the right side. The important contents of this label are a product-specific alphanumeric code and a scannable data matrix.

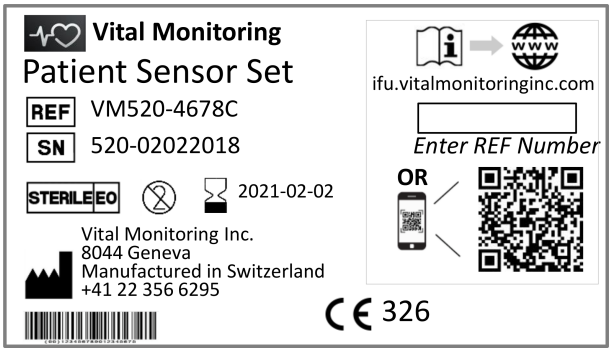

Figure 2. Example of a dokspot Label.

To access the instructions, the healthcare professional simply scans the data matrix (using for example a standard QR code reader app on a smartphone or tablet) or enters the product-specific code (here: VM520-4678C) on the web page at the URL provided by the manufacturer (here: ifu.vitalmonitoringinc.com). As a result, the instructions are displayed in the required language.

It is obvious that this approach removes some of the limitations identified earlier. However, there is still the challenging problem of how such a service should be designed and developed so that it provides sufficient protection from attacks. In particular, it must be guaranteed that when a user accesses online instructions through dokspot, there is high assurance that the instructions they receive and view on their device match the original instructions as provided by the manufacturer of the healthcare product. In this context, it is also important to realize that several regulatory authorities have issued guidelines obliging manufacturers to protect the integrity of their electronic files, which includes protecting online instructions from unauthorized modification. One recent example for such a guideline is the GXP Data Integrity Definitions and Guidance from the UK Medicine & Healthcare Products Regulatory Agency (MHRA) [1].

To our knowledge, no work has been published so far that presents a solution to this challenging problem. The focus of this paper therefore lies on filling this gap by presenting the underlying security architecture of dokspot. This security architecture contains several key components to protect from a wide range of attacks and provides the basis to achieve the required level of security and also to comply with regulations as mentioned earlier. As a result of this, dokspot truly allows to securely link healthcare products with online instructions. Note that while this security architecture has been designed, developed and evaluated in the context of dokspot and for healthcare products, the architecture is general enough so it can be applied to other industries as well. Therefore, the main contribution of this paper is to provide a security architecture that allows to securely link physical products and online documents in general, independent of the field of application.

The remainder of this paper is organized as follows: Section II describes the security goals and threat model of dokspot. Based on this, the security architecture is explained in Section III, followed by the evaluation in Section IV. Related work is covered in Section V and Section VI concludes this work.

II. Security Goals and Threat Model

In this section, the overall architecture, the security goals and the threats against dokspot are described. This provides the basis for the security architecture that follows in Section III.

A. Overall Architecture

Figure 3 depicts a high-level view of the overall architecture of dokspot.

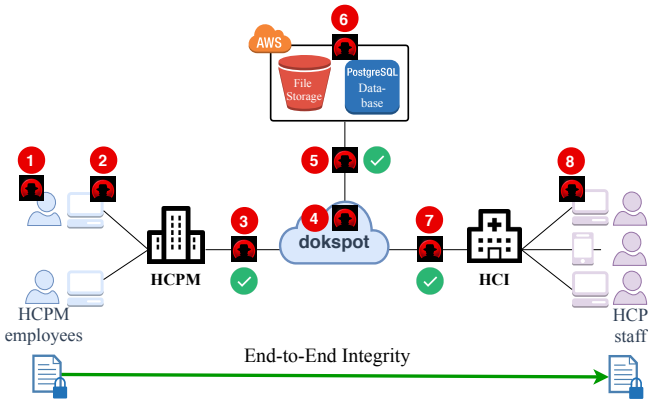

Figure 3. dokspot Architecture and Threats against dokspot.

In the middle of Figure 3, there is the central component of the dokspot service. This component corresponds to a web application that provides the core functionality and which is deployed on a suitable cloud application platform. To store the instructions, a cloud storage provider is used. Using 3rd party services to run the application is much more reasonable than deploying and operating an own infrastructure, especially as these services provide high availability and can easily scale with growing demands with respect to computing power and storage space.

On the left side, there is a company – identified as healthcare product manufacturer (HCPM) – that uses dokspot to provide its instructions. HCPM employees are interacting with the dokspot service via the web browser on their

workplace computers to upload and maintain instructions of their products. To support this, the dokspot service provides a corresponding web interface.

On the right side, there is a healthcare institution (HCI) that is using the online instructions provided by the HCPMs via dokspot. Its users are identified as healthcare professionals (HCP). HCPs can access the instructions using any web browser on any device, including workstations, laptops, tablets or smartphones. This is also supported with a corresponding web interface provided by the dokspot service.

B. Security Goals

With respect to the general security goals – confidentiality, integrity, and availability – dokspot is a "special case". In many typical Internet-based applications, confidentiality is highly important, but in the case of dokspot, confidentiality of the instructions is not critical at all. All instructions provided through dokspot are usually considered public information and it is therefore not necessary to restrict read-access to them.

Availability and integrity are paramount, though. The instructions should be available within a few seconds whenever they are needed by a HCP and unavailability may mean – in the worst case – that patient safety is negatively affected. High availability is achieved by operating the service using well-established cloud-based services that have demonstrated to be suitable for high-availability services and by putting a strong focus on robust and secure software during the entire development process.

Integrity is even more critical. Tampering with medical device instructions without anyone detecting this can have a severe impact on patient safety, which in turn would likely result in legal consequences for all involved parties. E.g., a deliberately wrong dosage of a substance, an altered configuration value of a life-sustaining machine or misinformation about allergens present in a product can lead to a serious endangerment of patients up to fatal complications. A lack of integrity protection can be exploited in a variety of attacks, e.g., by a disgruntled employee at the HCPM to harm their employer, by an external attacker to blackmail the HCPM or the dokspot service provider (request them pay a sum of money, otherwise tampering with the instructions will begin or continue), by a competitor of the HCPM, dokspot service provider or the HCI to gain an own advantage, and so on.

From this discussion, it follows that providing end-to-end integrity protection is crucial to achieve a truly trustworthy linking of physical products with online instructions. End-to-end integrity means that the HCP receives the original instruction that was provided by the HCPM, i.e., if the HCP receives and views an instruction on their device, then there is high assurance the instruction can be trusted and has not been tampered with. The bottom part of Figure 3 illustrates this end-to-end integrity: A HCPM employee provides an instruction and the dokspot service must make sure that the instruction is delivered to the HCP in the original form.

In the remainder of this paper, the focus will be on achieving this integrity-protection. Of course, other security aspects are relevant as well, but they can be solved by using state-of-the-art technologies and practices (as described, e.g., in [2]) and therefore do not require innovative approaches to be solved.

C. Threat Model

Figure 3 also includes possible threats against dokspot to compromise the integrity of instructions. The corresponding attack points are numbered 1–8 and explained in the following list.

- **Attack point 1**: A HCPM employee uploads manipulated instructions, either because they want to harm any of the involved parties or because they were bribed by someone else. Such internal attacks are common and their percentage among all cyber attacks is rising [3]. Note that the dokspot service provider could argue that this is out of scope: it is the problem of the HCPM to make sure no tampered instructions are uploaded in the first place. However, this would contradict the goal of being a truly trustworthy service, therefore dokspot should provide secure workflows that at least significantly increase the difficulty of such an insider attack.
- **Attack point 2**: The computer or web browser used by a HCPM employee to interact with the dokspot service could be compromised by an attacker. There are different attack vectors to achieve this, the most common ones include malicious e-mail attachments, drive-by downloads via compromised websites, and infected media (often USB sticks).
- **Attack points 3, 5, 7**: An attacker that has access to any of the communication channels can modify the transmitted instructions at will. This is a communication security problem and there exist good standard solutions to solve this problem by employing secure communication protocols, typically Transport Layer Security (TLS) [4]. Therefore, this attack point will not be addressed further in the remainder of this paper.
- **Attack point 4**: The attacker could get illegitimate access to the dokspot service, either by guessing or stealing (e.g., with a social engineering attack) the credentials of a HCPM employee or by directly compromising the service. While getting the credentials of users can effectively be prevented using strong authentication methods, it is much harder to make sure that the service cannot be compromised by exploiting a vulnerability. Putting a strong focus on secure software development can significantly reduce the risk of critical vulnerabilities, but today, there are no practical methods that can guarantee a 100% vulnerability-free service. The dokspot service primarily provides a web application interface, where security is especially hard to achieve: According to the Website Security Statistics Report of WhiteHat Security [5], 86% of several 10'000 analyzed websites contained at least one highly critical vulnerability. Therefore, the security architecture of dokspot should allow to guarantee integrity of instructions even if the service is (partly) compromised.
- **Attack point 6**: The storage service is another point of attack. While renowned companies are most likely taking great care with appropriate security measures, it is nevertheless possible that instructions can be manipulated while being stored in the storage service, either be external or internal attackers. Therefore, the

 ISBN: 978-1-61208-675-0

dokspot service should be able to cope with such attacks by making sure that modified instructions can be detected.

- **Attack point 8**: Just like with attack point 2, the computers, web browsers, or mobile devices used on the side of the HCI could also be compromised. This allows an attacker to exchange the requested instruction with any instruction the attacker wishes. In contrast to "standard" computers, mobile devices implement a stronger security model and therefore, attacks against mobile devices are much more difficult to execute and therefore occur significantly less frequently [15]. Most malware incidents on mobile devices happen because users install apps from untrusted sources. However, such malware is then confined to the actual app (due to the sandboxing model implemented by mobile devices) and can neither affect the underlying operating systems nor other apps (and also not the web browser). As a result, the risk of powerful malware on mobile devices that can affect the instruction that is requested and viewed in the web browser is small.

To summarize, there exist well-established security measures to secure (and integrity-protect) communications channels in the Internet. Therefore, attack points 3, 5, and 7 are marked with a green "solved mark" in Figure 3. In addition, mobile devices provide good protection against powerful attacks by design, so it is unlikely that they are compromised. With respect to all other attack points, however, there are no available standard solutions that could be applied.

III. Security Architecture

In this section, key components of the security architecture of dokspot are described. First, a secure workflow to manage online instructions is introduced. Next, the microservices-based approach of dokspot is described, followed by a brief introduction of the cloud infrastructure that is used by dokspot as a basis. After that, the usage of digital signatures to integrity-protect instructions during the entire lifecycle is explained before the section is completed by describing some further, more common security measures that are employed.

A. Secure Workflow to Publish Instructions

The goal of this workflow is to securely publish instructions and therefore make them publicly available to HCPs. To mitigate risks originating from a rogue HCPM employee (attack point 1), a compromised computer (attack point 2) or illegitimate access to the dokspot service (attack point 4), a workflow based on a segregation of duties (SoD) approach was introduced. The main idea is to split the process of publishing an instruction into several steps that are only executable by different authorized HCPM employees. To do this, Role-Based Access Control (RBAC) is used as the authorization mechanism [6] and the workflow enforces that at least three different roles must be involved in order to publish an instruction. As an employee can typically only have one of the three roles, this ensures that a minimum of three employees are required to publish an instruction. As a result of this, a single employee (and also two colluding employees) is never empowered to execute the entire workflow and is therefore unable to publish malicious instructions. Figure 4 illustrates the basic idea of this workflow.

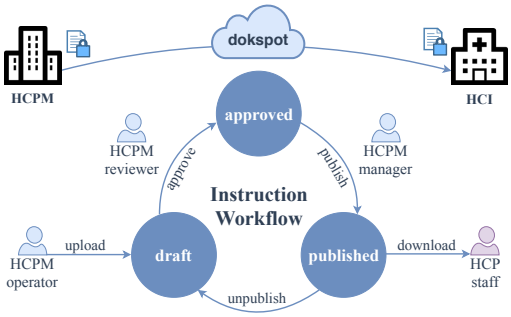

Figure 4. Workflow to Publish Instructions

As can be seen in Figure 4, an HCPM employee that has the role *operator* uploads an instruction to the dokspot service. As a result, the newly introduced instruction gets stored in state *draft* and needs to be approved by one or more HCPM employees with the role *reviewer*. A successful review process puts the instruction into the state *approved*. It can now be published by the HCPM manager of the corresponding healthcare product (corresponds to employees with the role *manager*). This results in a change of the state of the instruction to *published*, which means it is now publicly available and can be downloaded and viewed by the HCPs via the dokspot service.

This secure workflow prevents several attacks. For instance, if a malicious employee with role *operator* uploads a malicious instruction, this will most likely be detected by the employees with role *reviewer* as it is their obligation to review the content of the instructions for correctness. Likewise, it may be that malware on the computer of the *operator* modifies the instruction before it is uploaded, but just like in the first case, this should be detected by the *reviewers*. Note also that employees with roles *reviewer* and *manager* can only change the state of an instruction, but not its content, so it is not possible that one of them (or their computers, in case they are compromised) can modify an instruction in a malicious way.

B. Microservices

As mentioned earlier, the central component of the dokspot service is a web application. Nowadays, when developing a modern web application, developers heavily rely on pre-existing libraries. This includes frameworks that usually already consists of hundreds of thousands lines of code, plug-ins that themselves depend on many other plug-ins, middleware that helps to glue together all the pieces of a modern cloud-based architecture, and more. Unfortunately, the more complex a web application gets, the higher are the chances that one of its components has a security flaw that can be exploited by an attacker. That means that even if the developer of the web application itself is able to produce secure code (which is hard), there might still be some vulnerabilities lurking in the numerous dependencies, over which the developer has little control.

In addition, web applications often use monolithic architectures, which means there is usually one server (or multiple in the case of load-balancing) that is capable of handling all incoming requests. This implies that this server has to contain all of the required software components (which includes, as

 ISBN: 978-1-61208-675-0

mentioned earlier, application, framework, plug-ins, middleware, etc.) at some point and must have full access to all data that is processed by the application. This leads to the unfavorable situation that attackers have many potential points of attack available to break into the system, and once they succeed in doing so, they are typically able to access and manipulate all data available to the web application.

To mitigate this risk, the dokspot service was split into multiple sub- or microservices, which handle just specific parts of the entire functionality. As a result of this, individual parts of the application, especially parts that handle sensitive data (e.g., login information), can be hardened against attacks, e.g., by blocking all requests outside their area of responsibility. Also, such a microservice requires just a small subset of the entire codebase and is therefore much harder to attack. Furthermore, the different microservices run on different servers with restricted access to the storage subsystems. This implies that if an attacker manages to get unauthorized access to one of the microservices, their possibilities are limited by the boundaries of the specific capabilities of the compromised service.

How to split up an application into microservices is dependent on the actual application. In the case of the dokspot service, this resulted in the microservices shown in Figure 5.

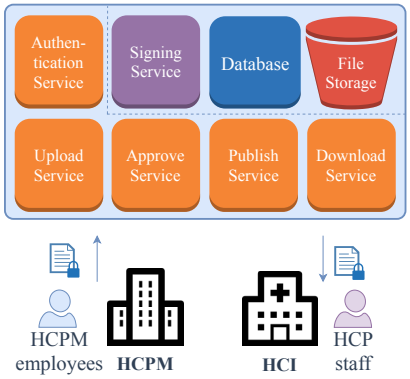

Figure 5. dokspot Microservices

Figure 5 shows the six microservices (in short *service*): authentication, upload, approve, publish, download and signing as well as the two storage types: database and file storage. Instructions are stored in the file storage, whereas all other data is stored in the database. The access privileges of the different services to the database and the file storage are restricted to the bare minimum, meaning that each service is only allowed to access or modify information which is required by the service in order to work as designed.

The authentication service is used to authenticate HCPM employees during login and to handle other tasks related to user management. The authentication service has exclusive access permissions to security-relevant data in the database such as the passwords of HCPM employees or their assigned roles. This also means that solely the authentication service is allowed to modify user accounts of HCPM employees and all modifications, such as password changes or role assignments, must happen via the authentication service. Naturally, due to its access to highly critical data, the authentication service is an attractive attack target, and therefore security was taken very seriously during its development. This included, e.g., working out a detailed security design as its basis, performing thorough code reviews and doing penetration tests. These are usually quiet costly activities, but due to the microservices-based architecture, they could be done in the context of a component with limited complexity, which allows to carry them out efficiently and which significantly increases the probability that the outcome is secure.

The upload, approve and publish services provide the functionality to perform the different steps during the workflow to publish instructions (as illustrated in Figure 4). They are only accessible by HCPM employees that have the corresponding role and only after they have successfully authenticated at the authentication service. This means the upload service is only accessible by authenticated HCPM operators, for the single purpose to upload instructions to dokspot. The upload service has exclusive rights to store instructions in the file storage and permissions to insert the corresponding metadata into the database. Once uploaded, an instruction can neither be modified nor deleted by the upload service or any other service, because the file storage does not allow it. Therefore, compromising any service does not allow an attacker to delete instructions. The approve service is used by authenticated HCPM reviewers and enables them to approve instructions after reviewing them carefully. The publish service is used by authenticated HCPM managers to make approved instructions publicly available.

The advantage of having different services to upload, approve and publish instructions becomes apparent under attack. For instance, a compromised upload service enables an attacker to upload malicious instructions with wrong or harmful content, but due to the service architecture, it is not possible to publish the instruction without compromising the approve and publish services as well. The reason is that the upload service lacks the capabilities and permissions on the database to complete the review or publish steps. On the other hand, if an attacker controls the publish service, it is still not possible to make harmful instructions publicly available, because the service is missing the upload functionality and permissions. To summarize, it is required to compromise all three services in order to publish an arbitrary instruction.

The download service is publicly accessible, without the need to authenticate, and is primarily used by HCPs. Nevertheless, the download service is critical, because if an attacker manages to compromise it, they can basically serve any instructions they like to the HCPs. Therefore, just like the authentication service, this service is also especially hardened and tested. As the service contains only relatively little functionality and only serves one single purpose, this was possible with a reasonable amount of resources.

The signing service is an internal service, which means it can only be accessed by some of the other services, but it is not accessible (and also not visible) from the Internet. This service is used to digitally sign instructions, which is explained in detail in Section III-D. The reason why this service is separated from the others and not publicly reachable is because it contains sensitive private keys used for signing, which must not fall into the hands of an attacker. If an attacker manages to get access to this key material, then they can publish fraudulent instructions, assuming they also get access to the database and the file storage. As an internal service, this service is considered difficult to compromise, because to start trying to

 ISBN: 978-1-61208-675-0

attack it, an attacker first has to successfully compromise any of the other services. Or to put it differently: An attacker would be required to deeply infiltrate the dokspot service in order to reach the secrets to digitally sign instructions.

To summarize, the microservices-based architecture has several security benefits. First of all, the complexity of each service is much smaller compared to a monolithic approach, which reduces the attack surface of each individual service and which makes it easier to design, develop and configure them in a secure way. In addition, it reduces the impact of an attack, as in many cases, overall security is still maintained even if an attacker manages to compromise one of the services. And finally, it allows to hide services that provide functionality that must not be made available to the users (and therefore also the attackers), which further increases protection.

C. Infrastructure

More and more companies that provide Internet-based services do this over infrastructure of commercial cloud platform providers. As running an adequate data center usually is not a core competency of most companies, renting computing and network capacity is often the best option to meet the requirements at a reasonable cost. However, this leads to some loss of control over the service, which implies that it is crucial to pick a reputable provider that can demonstrate its trustworthiness, e.g., by possessing compliance certificates of advisable standards. If this advice is followed, one typically gets a higher level of security than by hosting the application and all data in a self-owned but unprofessionally managed infrastructure.

Dokspot is hosted on Amazon Web Services (AWS). The microservices are running on Heroku, which itself is hosted on AWS Elastic Compute Cloud (EC2). The database consists of several PostgreSQL instances provided by Amazons Relational Database Service (RDS). Every microservice has to authenticate against the database with role-specific credentials to restrict access to tables, columns and rows. This ensures that each microservice can only read and alter the smallest possible subset of data necessary for its role. The file storage is using an Amazons Simple Storage Service (S3) Bucket. To provide fast and reliable delivery CloudFlare is used as a Content Delivery Network (CDN) and Domain Name System (DNS) provider for all the microservices.

D. Digital Signatures

Usually, web applications guarantee the integrity of their data by carefully crafting the business logic in a way that does not allow for unwanted manipulation by the users. If there is a need to trace the changes that happen to data (e.g., to get an audit trail), some kind of logging mechanism is typically implemented. Unfortunately, there are two major weaknesses with this approach: a) One can never be sure that the code that handles the business logic is free from errors, and b) an administrator with sufficient access rights to the back-end of the application can often alter data and logs in an untraceable fashion. This weakens the guarantees one can make about the integrity of the data, which in the case of dokspot would go against one of the main goals. To mitigate the risks of such manipulation, dokspot uses digital signatures to strengthen the auditability of relevant actions. Based on this, unauthorized modifications can easily be detected.

Every relevant action executed by a dokspot user (operator, reviewer or manager) is digitally signed with a user-specific signing key. This is done with public key cryptography using RSA [7], but more modern signature schemes with smaller signature and key sizes, e.g., ECDSA [8] or ED25519 [9], could be used as well. The signed data covers metadata such as the user-id, the executed action, the digest of an uploaded instruction, etc. To make strong claims about the expressiveness of such signatures, it is absolutely crucial that only the dokspot users themselves have access to their own private key. In particular, this implies that even a service administrator or an attacker gaining access to one of the application servers cannot access the private keys of the users, as this would enable them to produce valid digital signatures in the users' names.

To achieve this, the private key of a user is stored in encrypted form in the database and this encryption uses a secret key that is derived from a user-chosen password. The dokspot service never sees this password and the private key is only decrypted and used to create signatures in-memory on the client-side (i.e., within the browser). Figure 6 illustrates how the key pair of a user is initially created.

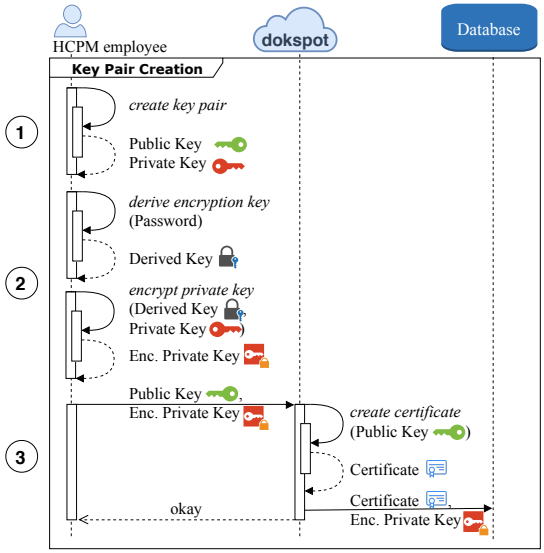

Figure 6. Key Pair Creation Sequence Diagram

In the first step (see Figure 6 (1)), after the first successful login, the user creates its own public/private key pair in the browser (using JavaScript code). Next (see Figure 6 (2)), they choose a dedicated password to protect the private key. To make brute forcing the password much more complicated in case an attacker gets access to an encrypted private key of a user, the Password-Based Key Derivation Function 2 (PBKDF2) [10] is utilized to derive a secret key based on the password. This secret key is then used to encrypt the private key. Once this has been done, the browser uploads the encrypted private key and the corresponding public key to dokspot (see Figure 6 (3)). Based on the received public key, the dokspot service then creates an X.509 certificate [11] and as a result of this, the key pair is now certified and can be used for signing and for verification of corresponding signatures in the context of the dokspot service. Finally, both the certificate and the encrypted private key are stored in the database.

 ISBN: 978-1-61208-675-0

Once this has been completed, the user can create signatures, e.g., to sign instructions during the upload step. This process is illustrated in Figure 7.

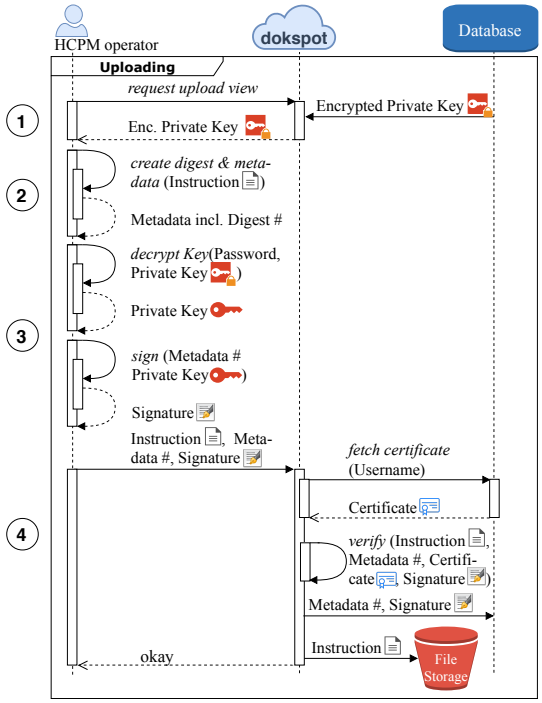

Figure 7. Instruction Uploading Sequence Diagram

To upload an instruction, the user first navigates to the corresponding view of the dokspot service by using their web browser. Before sending the web page to the browser, the dokspot service fetches the user's encrypted private key from the database and embeds it into the page (see Figure 7 (1)). Next, the user picks the instruction they want to upload and enters the associated metadata (title, language, etc.) (see Figure 7 (2)). In the background, a JavaScript function starts calculating the digest for the chosen instruction, using a cryptographic hash function. Once the user has finished entering the metadata, they start the signing process (see Figure 7 (3)) by clicking a button. The user is asked to enter the password that was used to protect their signing key and if the password is correct, the private key is decrypted. A JavaScript routine then adds a timestamp and the digest to the metadata and signs the resulting metadata with the private key.

Next, the instruction, the metadata, and the signature are sent to the dokspot service (see Figure 7 (4)). Upon receiving the data, the service fetches the certificate of the current user from the database and uses the public key in the certificate to verify the signature. If it is correct, it stores the instruction in the file storage and the metadata including the signature in the database. As a result of this, the signature now seals both the instruction and the metadata. If an attacker manages to alter just a single bit in this instruction later during the lifecycle of the instruction, this can easily be detected as verifying the signature will fail.

When a user with the appropriate role changes the state of an instruction (e.g., from *draft* to *approved*, see Figure 4) then this action also results in creating a digital signature. Technically, this works similar as as in Figure 7, meaning that the user navigates to the corresponding view, which again includes the user's encrypted private key. As soon as the user wants to initiate the state change (e.g., after they have checked the instruction and made sure it can be approved), they click a button, which triggers the change of the state: In a first step, further metadata is produced, which includes the specific action to be performed (e.g., the approval of the instruction) and a timestamp. Next, the digest of the instruction is included in the metadata and the resulting metadata is signed with the user's private key (which, just like earlier, requires the user to enter their password). All of this is then sent to the dokspot service, which checks if the user is allowed to execute the specified action and which checks the validity of the signature. If all of the requirements are satisfied, the received metadata is stored in the database and the state of the instruction is updated in the database.

To further enhance the auditability, the dokspot service adds its own signatures to important actions. So when the service receives an uploaded instruction or a state change, a second signature, which acknowledges the reception, is created and saved along with the metadata and the signature of the user. This additional signature serves as a proof that the dokspot service verified the user's signature of the action and that is has been declared valid.

When a HCP wants to access an instruction, they have to navigate to the appropriate view, which shows the published instructions for a specific product. After the user picks the instruction they want to read, the dokspot service collects all the signed metadata attributed to this instruction. The service then checks if all the necessary signatures are available and valid, i.e., it is checked whether the instruction contains valid signatures to verify it was uploaded, it was approved, it was published and so on. To do these checks, the service freshly calculates the digest of the requested instruction and cross-checks it with the digest specified in the signed metadata. If any of the checks fail, the instruction will not be delivered to the HCP as this is an indication of an attack.

E. Passwordless Signatures

So far, it is assumed the users use a dedicated password to protect their private key. This mitigates a range of attacks, as this password will never be sent across the network. However, it requires that the user has to remember a second password (in addition to the login password). Also, they have to enter the password that protects the private key every time a signature should be created. While this is the most secure configuration, it is also somewhat inconvenient. For this reason, a password-less signing process was developed, which lets a user trade some of the security for a more convenient experience of the dokspot service.

With passwordless signing, the login password is used to encrypt the private key. After the user has entered this password during login (which they have to do anyway when using dokspot), the password is temporarily stored so it can be used to automatically decrypt the private key at a later point. Storing the password in plain text on the client side would not be ideal from a security perspective as it may provide an attacker that somehow gets access to the system with an opportunity to extract this password. Therefore, the password is protected with the approach illustrated in Figure 8.

 ISBN: 978-1-61208-675-0

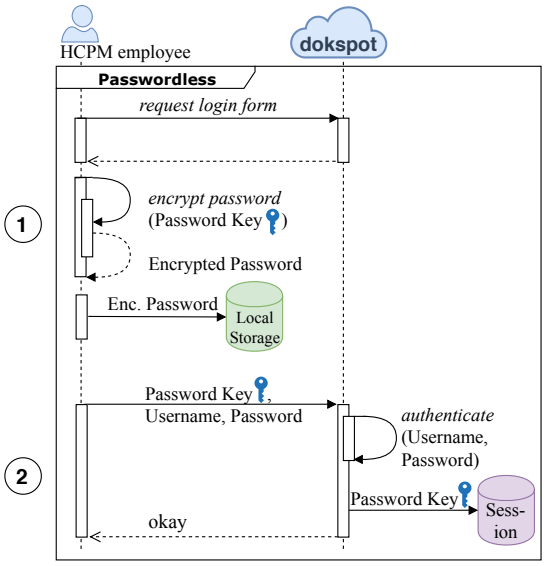

Figure 8. Passwordless Signing Sequence Diagram

When the user opens the login page, a JavaScript function in the background choses a sufficiently large symmetric key at random, identified as the password key (see Figure 8 (1)). This password key is stored in a hidden field in the login form. When the user sends the login form with the password key to the dokspot service, a JavaScript function in the browser encrypts the entered password with the password key and stores the encrypted password in the browsers local storage. If the authentication is successful, the service stores the received password key in the session of the user (see Figure 8 (2)). This means that once the login is done, both sides hold just one piece of the information needed to decrypt the user's password. The client needs the password key from the service to decrypt the locally stored encrypted password and the service would need the encrypted password to do the same. That means if an attacker gains access to only the client (or only the service), they will not be able to retrieve the password.

Note that during a login procedure with a simple POST request, the user's password must be exposed to the dokspot service. This implies that theoretically, the service could try to decrypt the user's private key at that point. This can be prevented by using a more secure login scheme, utilizing, e.g., the Secure Remote Password (SRP) Protocol [12].

The process of creating a passwordless signature is essentially the same as described earlier in Figure 7. This time, however, the user does not only get the encrypted private key from the dokspot service, but additionally also the symmetric password key. Next, a JavaScript function in the browser will use the password key to decrypt the encrypted password that is stored in the local storage. And finally, the decrypted password can be used to decrypt the private key, which can then be used to create digital signatures.

The confidentiality of the user's password is not affected by this signing scheme. Independent of entering the password manually or recovering it via JavaScript, the password is accessible in the JavaScript runtime environment during the signing process either way.

F. Common Security Measures

Besides the very specific and innovative security measures described earlier, the dokspot service also employs several state-of-the-art security measures, some of which are briefly summarized in this paragraph. First of all, the dokspot service can only be reached using HTTP over TLS (HTTPS). This provides an encrypted and integrity-protected communication channel. An HTTP Strict Transport Security (HSTS) policy is in place and preloaded, to further increase the difficulty of an attack against the connection. The service uses secure cookies (encrypted and signed), a Content Security Policy (CSP) and cross-site scripting protection to reduce the risk of a break-in. The DNS entries of the dokspot domain are protected by DNSSEC [13] and DNS Certification Authority Authorization (CAA) to make it difficult for an attacker to reroute users to a fake service. The login, located on the authentication service, can be secured using two-factor authentication. This mitigates the risks of a stolen, lost or phished password. In addition, some HCPM employees (e.g., the manager of an entire product family), can be notified if a suspicious or important events, such as publishing an instruction to the public, takes place. This allows to quickly react in case of a potential security breach. Finally, to verify the high level of security, the dokspot service has been tested for vulnerabilities by security professionals.

IV. Evaluation

Coming back to the threat model in Section II-C, we can see that the risks in the context of the identified attack points could be mitigated, except for attack point 8.

If, due to attack points 1 or 2, a non-genuine instruction gets uploaded onto the dokspot service, this cannot be detected automatically because the platform itself is oblivious of the content of uploaded instructions. The workflow, however, enforces an instruction to be reviewed by multiple parties before it can be published to the public audience. Assuming that at least one of the involved parties performs their part of the process diligently, a forged instruction will never be made accessible to the HCPs.

To mitigate the risk of a break-in into one of the servers, as mentioned in attack point 4, the service is split into multiple microservices and all relevant actions are digitally signed. The microservices that handle the processes of uploading, approving and publishing instructions are the ones with the widest range of functionality. They are therefore the most probable candidates to contain vulnerabilities that can be abused to gain illegitimate access to the dokspot service. However, an attacker with access to any subset of these exposed microservices will not be able to produce the full valid set of signatures required for an instruction in state *published*. As the download service checks the full set of signatures before an instruction is delivered to an HCP, it will not be possible for an attacker to provide manipulated instructions through dokspot.

An exposed point in this scenario is the download microservice. An attacker controlling this component can deliver whatever instructions they wish to HCPs because it allows the attacker to bypass any form of signature validation. But as the download microservice just offers a single and very simple functionality, it is relatively easy to harden this service to a point where a successful break-in is very unlikely.

 ISBN: 978-1-61208-675-0

In attack point 6, a breach of the storage system is assumed. As Amazon RDS and S3 are used for storage, we will not discuss how an attacker might achieve this and focus on the consequences. The signed metadata described in Section III-D holds, among other information, the digest of the corresponding instruction. Dokspot regularly verifies that the digest of the instructions matches the digests that are stored as part of the signed metadata. This means that if any instruction – even a single letter – were modified for whatever reason, this would be detected during such a check. Also, as mentioned earlier, the full set of signatures is checked whenever an instruction is requested by an HCP, where any manipulation would be detected as well. To summarize, this means that any kind of breach of the storage system cannot result in delivering a manipulated instruction to the HCPs.

Unfortunately, attack point 8 remains an open problem. Assuming the attacker has control over the device used by an HCP to download an instruction, there is nothing the dokspot service can do to prevent the attacker from displaying whatever instruction they wish. Therefore, preventing such attacks is currently out of scope for dokspot and it is the responsibility of the HCI to make sure its IT infrastructure is malware-free.

V. Related Work

Microservices recently became popular as a service architecture in the web environment [14] and provide, compared to a monolithic approach, benefits, such as scalability, cost reduction and improved performance [15]. While some papers discuss the security challenges of microservices, such as authentication or communication between services [16], as well as auditability or inter-service trust [17], they do not highlight the security benefits. Therefore, using microservices to increase application security – as done with the dokspot service – appears to be a novel approach.

The idea of using digital signatures to safeguard business workflows has been covered in multiple papers since 1999 (e.g., [18], [19]). However, previous works do not cover the particular characteristics that must be considered when using signatures in modern web applications or role-based microservices. In the domain of user signatures, Halpin wrote a critical acclaim of the W3C web cryptography API (and browser-based cryptography in general) [20], which highlights some limitations that come with the fact that the web server is mostly in control over the executed JavaScript code. To address this, our approach with SoD for HCPM employees and microservices makes it very difficult for an attacker to gain control over the necessary key material to create a valid set of signatures.

VI. Conclusion

In this paper, we presented dokspot, which is an Internet-based service to securely link physical products with online instructions. To achieve this, dokspot is based on a sophisticated security architecture that combines several approaches. We have shown that with the help of a well-tailored workflow, strong claims can be made about the genuineness of instructions managed by the dokspot service. By utilizing digital signatures on multiple layers, the compliance of the workflow can be proven cryptographically and the integrity of instructions can be guaranteed. By appropriately splitting the service into microservices, the dokspot service gets hardened as a whole. Due to restrictions with respect to the access permissions of the microservices and the use of digital signatures, the possibilities for an attacker are greatly restrained, even in the case of a successful partial break-in. Finally, we introduced a passwordless signature scheme, which leads to a more convenient user experience when creating digital signatures without significantly reducing security. Overall, the presented security architecture makes dokspot highly resistant to a wide range of attacks.

Acknowledgement

This work was partly funded by the Swiss Confederation's innovation promotion agency CTI (project 18990.1 PFES-ES).

References

[1] MHRA, "GXP Data Integrity Guidance and Definitions," MHRA, Revision 1, Mar. 2018.

[2] OWASP, "OWASP Guide Project," 2014. [Online]. Available: https://www.owasp.org/index.php/OWASP_Guide_Project 2018.08.08.

[3] D. M. Upton and S. Creese, "The Danger from Within." *Harvard business review*, vol. 92, no. 9, pp. 94–101, 2014.

[4] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Requests for Comments, RFC Editor, RFC 5246, Aug. 2008. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5246.txt 2018.08.08.

[5] WhiteHat Security, "Website Security Statistics Report," May 2013. [Online]. Available: https://www.whitehatsec.com/wp-content/uploads/2013/05/WPstatsReport_052013.pdf 2018.08.08.

[6] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996.

[7] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2," Internet Requests for Comments, RFC Editor, RFC 8017, Nov. 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8017.txt 2018.08.08.

[8] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[9] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-Speed High-Security Signatures," *JCEN*, vol. 2, no. 2, pp. 77–89, 2012.

[10] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0," Internet Requests for Comments, RFC Editor, RFC 2898, Sep. 2000. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2898.txt 2018.08.08.

[11] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," Internet Requests for Comments, RFC Editor, RFC 5280, May 2008. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5280.txt 2018.08.08.

[12] T. D. Wu, "The Secure Remote Password Protocol." in *NDSS*, vol. 98, San Diego, CA, USA, Mar. 1998, pp. 97–111.

[13] M. Larson, D. Massey, S. Rose, R. Arends, and R. Austein, "DNS Security Introduction and Requirements," Internet Requests for Comments, RFC Editor, RFC 4022, Mar. 2005. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4033.txt 2018.08.08.

[14] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 116–116, Jan. 2015.

[15] M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, and S. Gil, "Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud," in *2015 10th Computing Colombian Conference (10CCC)*, Sep. 2015, pp. 583–590.

[16] N. Alshuqayran, N. Ali, and R. Evans, "A Systematic Mapping Study in Microservice Architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, Nov. 2016, pp. 44–51.

 ISBN: 978-1-61208-675-0

[17] Y. Sun, S. Nanda, and T. Jaeger, "Security-as-a-Service for Microservices-Based Cloud Applications," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, Nov. 2015, pp. 50–57.

[18] K. R. P. H. Leung and L. C. K. Hui, "Signature Management in Workflow Systems," in *23rd International Computer Software and Applications Conference*. Washington, DC, USA: IEEE Computer Society, Oct. 1999, pp. 424–429.

[19] H. W. Lim, F. Kerschbaum, and H. Wang, "Workflow Signatures for Business Process Compliance," *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 5, pp. 756–769, Sep. 2012.

[20] H. Halpin, "The W3C Web Cryptography API: Motivation and Overview," in *Proceedings of the 23rd International Conference on World Wide Web*. Seoul, Korea: ACM, 2014, pp. 959–964.

 ISBN: 978-1-61208-675-0