Chapter 2

Malicious Pixels Using QR Codes as Attack Vector

Peter Kieseberg, Sebastian Schrittwieser, Manuel Leithner, Martin Mulazzani, Edgar Weippl, Lindsay Munroe, Mayank Sinha
SBA Research gGmbH, 1040 Vienna, Austria

[1stletterfirstname][lastname]@sba-research.org

This work examines QR codes and how they can be used to attack both human interaction and automated systems. As the encoded information is intended to be machine readable only, a human cannot distinguish between a valid and a maliciously manipulated QR code. While humans might fall for phishing attacks, automated readers are most likely vulnerable to well-known types of attacks where input data is not sanitized properly such as SQL and command injections. Our contribution consists of an analysis of the QR code as an attack vector, showing different attack strategies from the attackers point of view and exploring their possible consequences in a proof-of-concept phishing attack against QR codes, that is based on the idea of changing the content of a QR code by just turning white modules (pixels) into black ones.

2.1 Introduction

A QR ("quick response") code is a two dimensional barcode invented by the Japanese corporation Denso Wave. Information is encoded in both the vertical and horizontal direction, thus holding up to several hundred times more data than a traditional bar code (Figure 2.1). Data is accessed by taking a picture of the code using a camera (e.g. built into a smartphone) and processing the image with a QR code reader.

QR codes have rapidly gained international popularity and found widespread adoption, especially in Japan where its ability to encode Kanji symbols by default makes it especially suitable. Popular uses include storing URLs, addresses and various forms of data on posters, signs, business cards, public transport vehicles, etc. Indeed, this mechanism has a vast number of potential applications [1–5]. For instance, the sports brand Umbro has

I. Khalil and T. Mantoro (eds.), *Trustworthy Ubiquitous Computing*,
Atlantis Ambient and Pervasive Intelligence 6, DOI: 10.2991/978-94-91216-71-8_2,

Fig. 2.1 2D and 3D codes

embedded QR codes into the collars of England football shirts, sending fans to a secret website where prizes can be won.

In this chapter, we explore the structure and creation process of QR codes as well as potential attacks against or utilizing QR codes. We give an overview of the error correction capabilities and possible ways to alter both error correction data and payload in order to either modify or inject information into existing codes. Furthermore, we explore numerous vectors that might enable an attacker to exploit either the user's trust in the content embedded in the code or automated processes handling such codes.

Our main contributions are:

- to outline possible modifications to different parts of QR codes such as error correction codes or masking,
- to describe resulting attack vectors, both against humans (e.g. phishing attacks) and automated processes (e.g. SQL injections).
- to introduce a proof-of-concept phishing attack against QR codes.

2.2 Background

QR codes [6] have already overtaken the classical barcode in popularity in some areas. This stems in many cases from the fact that a typical barcode can only hold a maximum of 20 digits, whereas as QR code can hold up to 7,089 characters. Combined with the diversity and extendability offered, this makes the use of QR codes much more appealing than that of barcodes. Statistically, QR codes are capable of encoding the same amount of data in approximately one tenth the space of a traditional bar code. An important feature of QR codes is that they do not need to be scanned from one particular angle, as QR codes can be read regardless of their positioning. QR codes scanners are capable of determining

the correct way to decode the image due to the three specific squares that are positioned in the corners of the symbol and the alignment blocks.

QR codes were initially used by vehicle manufacturers for tracking parts. After a while, companies began to see the variety of different use cases for QR codes. The most popular commercial use for QR codes is in the telecommunications industry, where the increasing adoption of smartphones seems to be the biggest driver of their popularity [4, 7, 8]. With the technology of mobile phones constantly evolving, especially in the area of mobile internet access, QR codes seem to be an adequate tool to quickly and efficiently communicate URLs to users. This also allows offline media such as magazines, newspapers, business cards, public transport vehicles, signs, t-shirts or any other medium that can hold the print of a QR code to be used as carriers for advertisements for online products [9].

2.2.1 *QR codes*

There are 40 versions (sizes) of QR codes that consist of different areas that are reserved for specific purposes. In the following we refer to version 2 of QR codes (Figure 2.2), because version 1 does not contain all areas.

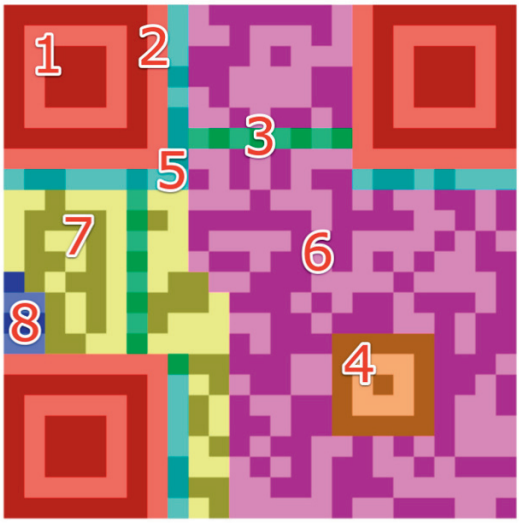

Fig. 2.2 Structure of QR code Version 2

- **Finder Pattern (1):** The finder pattern consists of three identical structures that are located in all corners of the QR code except from the bottom right one. Each pattern is based on a 3×3 matrix of black modules surrounded by white modules that are again surrounded by black modules. The Finder Patterns enable the decoder software to recognize the QR code and determine the correct orientation.
- **Separators (2):** The white separators have a width of one pixel and improve the recognizability of the Finder Patters as they separate them from the actual data.
- **Timing Pattern (3):** Alternating black and white modules in the Timing Pattern enable the decoder software to determine the width of a single module.
- **Alignment Patterns (4):** Alignment Patterns support the decoder software in compensating for moderate image distortions. Version 1 QR codes do not have Alignment Patterns. With growing size of the code, more Alignment Patterns are added.
- **Format Information (5):** The Formation Information section consists of 15 bits next to the separators and stores information about the error correction level of the QR code and the chosen masking pattern.
- **Data (6):** Data is converted into a bit stream and then stored in 8 bit parts (called codewords) in the data section.
- **Error Correction (7):** Similar to data codes, error correction codes are stored in 8 bit long codewords in the error correction section.
- **Remainder Bits (8):** This section consists of empty bits if data and error correction bits can not be divided into 8 bit codewords without remainder.

The entire QR code has to be surrounded by the so-called Quiet Zone, an area in the same color shade as white modules, to improve code recognition by the decoder software.

2.2.2 *Capacity and Error correction code*

The capacity of a QR code depends on several factors. Besides the version of the code that defines its size (number of modules), the chosen error correction level and the type of encoded data influence capacity.

- **Version:** The 40 different versions of QR codes mainly differ in the number of modules. Version 1 consists of 21×21 modules, up to 133 (lowest error correction level) of which can be used for storing encoded data. The largest QR code (Version 40) has a size of 177×177 modules and can store up to 23,648 data modules.

- **Error Correction Level:** Error correction in QR codes is based on Reed-Solomon Codes [10], a specific form of BCH error correction codes [11, 12]. There are four levels of error correction that can be chosen by the user at generation time. Higher error correction levels increase the percentage of codewords used for error correction and therefore decrease the amount of data that can be stored inside the code.
- **Encoded Data:** QR codes can use different data encodings (see Section 3.2.2 for detailed information on character encoding modes). Their complexity influences the amount of actual characters that can be stored inside the code. For example, a QR code version 2 with lowest error correction level can hold up to 77 numeric characters, but only 10 Kanji characters.

2.3 Security of QR Codes

2.3.1 *Threat Model*

One can distinguish two different threat models for manipulating QR codes. First, an attacker may invert any module, changing it either from black to white or the other way round. Second, a more restricted attacker can only change white modules to black and not vice versa.

2.3.1.1 *Both colors*

The easiest approach for attacking an existing QR code is by generating a sticker containing a QR code with the manipulated QR code in the same style as the original QR code and position it over the code on the advertisement. Of course this would either require some preparation or a mobile printer and design applications for a mobile device. At least when attacking on a large scale against one chosen target, the time needed for preparation should not pose a serious limitation.

Since this attack is trivial, we have decided to exclude it from the scope of this work. However, we believe that using this method in an attack against a real-world advertisement is a viable option for large-scale attacks.

2.3.1.2 *Single color*

In this case we restrict ourselves to the modification of a single color only. The background for this restriction lies in the scenario of an attacker seeking to modify a single poster on the fly just by using a pen (thereby reducing the possible modifications to chang-

ing white modules to black). This restriction is the basis for the attacks further outlined throughout this chapter.

2.3.2 *Attacking different parts*

Since QR codes contain a lot of different information, including meta information on version, maskings and source encoding, several different regions exist that can be targeted for the attack either individually or in combination.

2.3.2.1 *The masks*

Masks are used to generate QR codes with a even distribution of black and white modules (close to 50:50 and distributed well over the entire code). This increases the contrast of the picture and thus helps devices to decode it. According to the standard, when generating a QR code, every mask of the 8 specified ones is applied and each result is rated. The mask that results in the best distribution according to the rating is chosen. The effect of using correct masks can be seen in Figure 2.3. The left-hand side of the figure shows the distribution of black and white modules after applying the masking step based on the analysis of 20,000 randomly generated QR codes. The second graph shows the module distribution of the same 20,000 QR codes before masking, where, on average, the number of white modules (represented by the red curve) is approximately two-times the number of black modules.

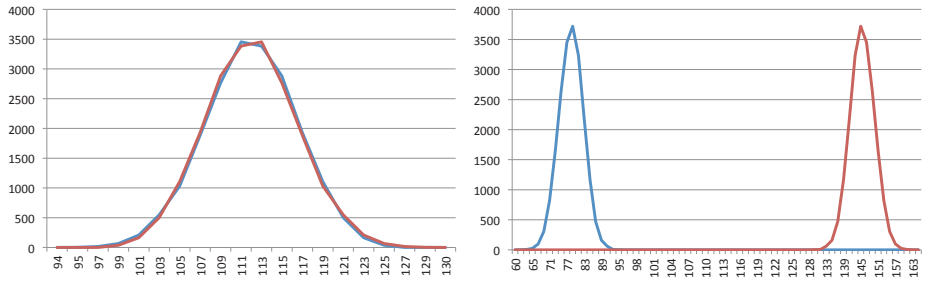

Fig. 2.3 Ratio of black and white modules with (left) and without (right) masking

There is always only one mask in use in a given QR code and it is encoded together with the version in a separate block of the code using strong BCH encoding.

Table 2.1 Mask Pattern References and conditions.

Mask Pattern	Condition
000	$(i+j) \bmod 2 = 0$
001	$i \bmod 2 = 0$
010	$j \bmod 3 = 0$
011	$(i+j) \bmod 3 = 0$
100	$((i/2)+(j/3)) \bmod 2 = 0$
101	$(ij) \bmod 2 + (ij) \bmod 3 = 0$
110	$((ij) \bmod 2 + (ij) \bmod 3) \bmod 2 = 0$
111	$((ij) \bmod 3 + (ij) \bmod 2) \bmod 2 = 0$

In the conditions in Table 2.1, i refers to the row position of the module and j to its column position. The mask is black for every module the condition is valid for and white for the rest.

Targeting the mask can change quite a lot in the whole data and error correction part, still, this can be a useful basis for additionally applying other methods. A problem when changing the masking is that it is encoded separately, utilizing a strong error correction algorithm.

2.3.2.2 *The character encoding (mode)*

There are several different source encodings (Table 2.2) specified for the information contained in the code, thereby maximizing the capacity in exchange for decreased complexity:

- Numeric mode (just encoding digits, thus being able to pack a lot of data in one picture),
- Alphanumeric mode (a set of characters containing upper case letters and several additional characters like *$* or *whitespace*),
- 8-bit mode (able to encode the JIS 8-bit character set (Latin and Kana) in accordance with JIS X 0201)
- Kanji characters (Shift JIS character set in accordance with JIS X 0208 Annex 1 Shift Coded Representation)

to name the most popular.

The character encoding itself is defined at the beginning of the data part by the leading 4 bits. Table 2.2 gives an overview on the possible values for the mode:

Changing the mode indicator gives the encoded data a whole new meaning. Especially when considering 8-bit Byte mode instead of other modes, or even alphanumeric mode in-

Table 2.2 Mode indicators.

indicator	mode
0001	Numeric mode
0010	Alphanumeric mode
0100	8-bit Byte mode
1000	Kanji mode
0011	Structured append
0101	FNC1 (first position)
1001	FNC1 (second position)
0000	Terminator

stead of e.g. Numeric mode, launching code injections (e.g. SQL injections) could become feasible (8-bit mode even allows for even more complex attacks using control characters).

An advanced attack against the mode can be mounted by mixing different modes in one QR code (see the section below).

2.3.2.3 *Character Count Indicator*

Right after the mode indicator, the next bits indicate the character count of the data that follows. The actual size of the character count indicator largely depends on the mode in use and the version of the QR code (higher versions contain more data, thus the character count indicator is longer). Refer to Table 2.3 for lengths for the most popular modes.

Table 2.3 Length of character count indicator.

Version	Numeric	Alphanumeric	8-bit	Kanji
1-9	10	9	8	8
10-26	12	11	16	10
27-40	14	13	16	12

Looking at this target we see two main approaches for an attack: Producing buffer overflows or buffer underflows.

- **Buffer underflow:** We change the character count indicator to resemble a lower number than in the original QR code. Thus, a decoding device should only decode the first few characters as message, leaving out the rest. This is especially useful in case the original link that was encoded contained suffixes. Since the size of the data part is fixed, everything after the anticipated number of bytes is either seen as a new segment (see mixed modes), or (in case of a terminator mode indicator) as filler. Since this is only a minor change in the data part (only the length is changed), this should in turn result in only a minor change in the error correction values and might still be decod-

able. Special attention should also be paid to a possible combination of this attack with other targets in case mixed modes are used.

- **Buffer overflow:** We change the character count indicator to a higher number, so the decoding device tries to decode parts of the filler as data (if we can even change the filler we gain valuable space for including additional data). Again, one of the drawbacks of this method lies in the error correcting (and especially error detecting) abilities of the Reed-Solomon-Code, which make it difficult to perform this attack in real-life scenarios.

2.3.2.4 *Mixing modes*

It is possible to use several different modes within one QR code (this is especially useful to increase density when encoding different types of data). To achieve this, several data segments with their own mode indicators and character count indicators simply get concatenated (see Figure 2.4).

Segment 1			Segment 2				Segment *n*		
Mode Indicator 1	Character Count Indicator 1	Data	Mode Indicator 2	Character Count Indicator 2	Data	...	Mode Indicator *n*	Character Count Indicator *n*	Data

Fig. 2.4 Message containing several modes

Again, we can identify four approaches of utilizing this feature of QR codes for attacks:

- **Changing modes of segments:** This works similar to an attack on the mode of a QR code that does not use mixed modes, but is reduced to one segment only, thus leaving other parts of the data untouched.
- **Inserting new segments:** Split an existing segment into two new segments, one with the old header (mode indicator and character count indicator) and one with a newly defined header.
- **Deleting existing segments:** Overwrite the space used by the mode indicator and the character count indicator of the segment with additional data, thus reducing the number of segments. Additionally the character count indicator of the segment before can be changed to allow the data of the deleted segment to be appended.
- **Structured Append:** The structured append mode allows for the concatenation of up to 16 QR code symbols, so a lot of data can be stored in this sequence of QR codes. It is designed in a way that decoding is independent from the order the symbols are read.

The basic idea of exploiting this lies in adding such a segment that points to another QR code that is stuck right beneath the original one. However, this attack requires stickers prepared in advance, making it less practical (it is far more work than just putting a prepared sticker on top of the original QR code, especially since the structured append mode is quite complex).

The main problem with all attacks against the mode (especially insertion and deletion of segments) is that they usually have a very high impact on the error correction codewords. Probably this attack can be used in combination with an attack on the error correction part itself.

Changing modes of existing segments should be attempted to be combined with color changes in the data section (and of course the error correction part), since it could be a valuable aid in case there are not enough white modules left for changing (remember that, as a prerequisite, we only consider changing one color to the other without the possibility to do the reverse). Also changing modes like numeric and alphanumeric to 8-bit would suddenly allow for unprintable control characters like *delete*.

2.3.2.5 *Data part and error correction*

The largest part of a QR code is made up of the parts containing data and error correction codewords. The data part itself can consist of segments using several different encodings, each with its own header specifying the mode in use and the length of the data following (see subsections on modes above). For a given version and error correction level, the part in the QR code that represents the data codewords and the part representing the error correction codewords can be defined easily without decoding, since the length of the data part is not depending on the actual length of the data (the data is filled up with padding patterns to the full length). For the position of the data part, refer to Figure 2.2. The exact length of the data part can be derived from the standard.

By design, any changes in the underlying data directly reflect on both the data and the error correction part. The Reed-Solomon encoding is able to detect several changes in either the data or the error correction part and provides decoding to the original message even after a moderate number of modifications/errors have been introduced, i.e. if Q_i denotes the 100% correct QR code for message M_i, then a QR codes Q_i' with only minor deviations to the original code Q_i can still be decoded back to M_i. This feature, while designed to protect the integrity of the code as much as possible, serves as an important prerequisite for our attack: We don't need to change the meaning of the original QR code Q_i to exactly

match the QR code $Q_j, i \neq j$ containing our manipulated QR code M_j, we just need to reach a code Q'_j that is decoded to the same message. In traditional computer security, this is analogous to NOP sleds that are used in buffer overflow attacks.

Figure 2.5 illustrates the outline of this attack. The green circles denote QR codes representing exact messages (i.e. containing no errors), the blue circles the set of erroneous QR codes that get decoded to the same message due to the error correcting features of the Reed-Solomon code. F denotes the set of QR codes that get detected as incorrect by the encoding but can not be corrected.

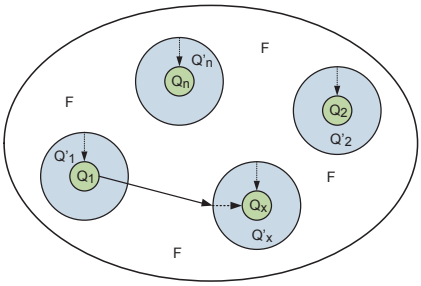

Fig. 2.5 Attacking data codewords

2.4 QR Codes as Attack Vectors

We believe that manipulated QR codes can be used for a plethora of attacks. Depending on whether the reader is a human or an automated program (e.g., in logistics), different scenarios are possible and outlined in this section.

2.4.1 *Attacking Automated Processes*

As QR codes are a standardized way of encoding information we strongly believe that the majority of software developers do not treat the encoded information as possibly insecure input. As described in detail in the previous section, different parts of the QR code could be manipulated in order to change the encoded information. Depending on the applications that process the encoded information, whether this would be in logistics, public transportation or in a fully automated assembly line, attacks on the reader software as well

as the backend are theoretically possible. Without proper sanitation this could by used by an adversary for the following, non-exhaustive list of attacks. Similar attacks using RFID chips and SQL injections have been shown to be very effective [13], as input sanitation was not employed in these examples.

- **SQL injection**: Many automated systems store and process the encoded information in a relational database. By appending a semicolon followed by a SQL query like `;drop table <tablename>` to the encoded information, manipulations to the backend database can be possible. This would delete the table specified in the command, resulting in a denial-of-service attack. More specific attacks may include adding a user, executing system commands (e.g., by using the stored procedure `xp_cmdshell` on Microsoft SQL Server), or altering data such as prices or passwords stored in the database.
- **Command injection**: If the encoded information is used as a command line parameter without being sanitized, this could be easily exploited to run arbitrary commands on behalf of the attacker, which may have disastrous consequences for the security of the operating system e.g., installing rootkits, DoS, or connecting a shell to a remote computer under the control of the attacker.
- **Fraud**: Changes to the automated system can be used to commit fraud, by tricking the system e.g., into believing that it processes a cheap product A while processing the more expensive product B.

2.4.2 *Attacking Human Interaction*

Humans can not read a QR code without a reader software, the information stored in the code is completely obfuscated for them. Hence, humans can not decide on the maliciousness of a code with decoding it with a reader software, whereby however a vulnerability in the application might get triggered.

- **Phishing and Pharming**: If QR codes are used for URLs in augmented reality scenarios, an attacker might set up a fake website and redirect users by changing the QR code. This is dangerous if some form of credentials are needed to access the website. The user has no possibility to verify that the URL is not modified. In section 2.5 this form of attack is demonstrated as a proof of concept.
- **Fraud**: QR codes are often used in advertisements to direct the target audience to special offers or additional information about specific products. If the QR code can

be manipulated to redirect the user to a cloned website, an adversary could sell the solicited product without ever fulfilling the contract. The victim implicitly trusts the advertising company by following the URL.

- **Attacking reader software**: Different implementations of the reader software on computers or smartphones might be attackable via command injection or traditional buffer overflows if the encoded information is not sanitized. An attacker might gain control over the entire smartphone, including contact information or the victim's communication data like e-mail or SMS.
- **Social engineering attacks**: Building on these attacks, more specific attacks like spear phishing or other variants of social engineering are possible, depending on the goal of the attacker. Leaving a poster of a QR code on the parking lot of a company (compare to the traditional attack with an USB drive) offering discount in a nearby restaurant is a new attack vector which is likely to be successful.

2.5 Proof-of-Concept Attack

In this section we propose a phishing-attack against printed QR codes, mainly considering codes placed in the public like on advertisements. Often these QR codes point to a website of the advertising company, thus being a valuable target for phishing.

An easy solution would be printing new QR codes (containing the phishing-address) on stickers and putting them on top of the original code. But after all, this can not be done unplanned, since every poster has got a different style, thus making stickers prepared in advance easily detectable. So we just want to use a simple device to devise the phishing-attack on the QR codes: A black pencil/marker. This results in the effect that changes of the QR code are reduced to changing white modules to black modules.

2.5.1 *Outline of the Attack*

Efficiently calculating similar code patterns is a difficult task, because the BCH codes produce unpredictable patterns in the error correction section of the QR code and we do not have influence on this part of the code. Our approach is to keep the data section as similar as possible (i.e. we make only changes in one codeword of the data section) and then calculate the number of changed modules in the resulting error correction section. This section describes the approach of the proposed attack. The vital steps are discussed

in detail below, the approach for a practical implementation is discussed in the following subsection.

(1) Scan the QR code (Q_0) with a mobile device capable of decoding QR codes and retrieve the corresponding Message M_0. For the rest of the paper we assume that M_0 is a URL to a website.
(2) Generate several messages $M_i,\ i = 1,\ldots,n$, that contain URLs to possible phishing sites (the new messages are generated in a way to make them look similar to the original one, e.g. by systematically changing characters in the original URL).
(3) Generate the corresponding QR codes Q_i for the messages $M_i,\ i = 1,\ldots,n$. The new QR codes should use the same version and mask as the original QR code, so no changes in these regions of the Code need to be done.
(4) Construct the symmetric difference D_i of the generated QR code to the original: $D_i = Q_0 \triangle Q_i,\ i = 1,\ldots,n$. The symmetric difference is defined as the set of modules on the same positions in their respective QR codes that differ in color.
(5) Calculate the ratios r_i of modules in the symmetric differences that indicate a change from white to black (thus fulfilling our initial condition):
$$r_i = \frac{|\{d_{i_j} \in D_i : d_{i_j} \ldots \text{ white}\}|}{|D_i|},$$
where d_{i_j} denotes the j^{th} module of D_i.
(6) Order the QR codes by ratio r_i, descending. Codes where the number of codewords (not modules) that need to get changed from black to white is higher than the error-correcting capacity of the code can be omitted.
(7) Start with the first QR code Q_1 (now sorted) and color white modules of Q_0 that are black in Q_1 black. Check after every module, whether the meaning of the QR code can be decoded and results in a different message than the original. Repeat this until a valid coloring is found (for the first b elements the check can be omitted, where b denotes the number of errors the BCH-encoding is capable of correcting plus one. If the resulting code Q_i' can get decoded to message M_i, a solution was found.
(8) The last step can be repeated for all Q_i where the number of black modules in the symmetric difference D_i is greater than the number of errors that can be corrected by the BCH-encoding (b).

In step two, the reason why we choose websites similar to the original one is that usually when a mobile device decodes a QR code, the content is displayed to the user. Thus the new message should look as unsuspicious as possible, i.e. the phishing must not be obvious

to heighten the success of our attack. Moreover we need the message to contain a valid internet domain that we can register for setting up the phishing-site. Using this approach we can validate the availability of the address before undergoing the more expensive task of trying to change the QR code to resemble it.

In practice we use the following optimization for steps four and five: Our QR codes Q_0 and Q_x are given in the form of matrices, where 1 denotes a black module and 0 a white module. The matrix symmetric difference can then be calculated using the element-wise *XOR*-function: $D_x = Q_0 \otimes Q_x$. In this matrix, a 1 denotes a field that needs to get changed and a 0 fields that contain the same color. The matrix R_x of the elements that need to get changed from white to black is defined by $R_x = (Q_x \wedge D_x)$, where $\wedge$ denotes the element-wise *AND*-function. The ratio r can then get calculated by using the 1-norm:

$$r_x = \frac{\|R_x\|_1}{\|D_x\|_1}$$

In step seven, the following optimization can be used: Instead of coloring module by module, we simply change all modules that can be changed by only using black color at once and thus generate Q'_x by applying the fast and simple element-wise OR-function: $Q'_x = Q_0 \vee R_x$.

2.5.2 *Practical application details*

To increase efficiency, the whole procedure should be placed in a small mobile application that automatizes the generation and validation of the new codes as much as possible.

The attacker scans the QR code Q_0 with his mobile device (e.g. a smartphone). Then the application on the device decodes the QR code and generates n messages $M_i,\ i = 1,\ldots,n$ containing similar domain names. It is important to choose a value n that is large enough to result in a good success-probability, but small enough to guarantee high run-times. Evaluating the size of this number with respect to the QR code parameters could pose an interesting question for future research. Furthermore, the application should check the web, whether these domains are available for registration, or not, in order to find a suitable phishing site. The QR codes Q_i are generated and rated. Thus the application iterates through all $Q_i,\ i = 1,\ldots,n$ and calculates the symmetric difference $D_i = Q_0 \triangle Q_i$ by XORing the respective matrices, as well as calculates the black/white-ration r_i of the resulting $D_i,\ \forall i = 1,\ldots,n$. Rating is achieved by ordering the Q_i in descending order according to their respective r_i. The application searches for the first suitable Q_i that will be decoded to a message $M_i \neq M_0$. The application then automatically registers the domain and displays

a detailed description to the user (i.e. in form of a matrix), how to change the original QR code Q_0 to the new phishing-version Q_i. Additionally the application could iterate through all M_i and display all possible solutions, allowing the attacker to choose one of them.

2.5.3 *Example*

We developed a proof-of-concept application that finds collisions to a given QR code based on the attack strategy outlined in the previous section. Figure 2.6 explains the attack based on the example URL `http://yahoo.at`. With only 20 white modules turned into black ones, the content of the QR code can be changed to the phishing URL `http://yghqo.at` which was not registered at the time we conducted this experiment. Given the small display size and the general trust of people towards automation, we are sure that a lot of people would fall for this phishing attack. In Figure 2.6 we show the original Q_0 (first square) for Message M_0=`http://yahoo.at`, a solution Q_i (third square) that is "corrected" to QR code Q_i (fourth square), encoding message M_i=`http://yahoo.at`, as well as the modules needing a black coloring in order to achieve this transformation (second square). Note that in this example, we do not have to turn any black modules into white ones in order to change the encoded message of the QR code.

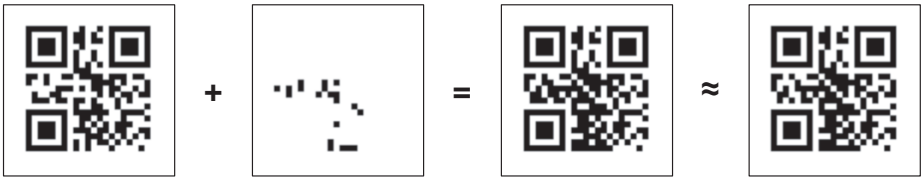

Fig. 2.6 Example phishing attack.

2.6 Future research

The possibilities for attacks proposed in this paper open up a quite large field for further research. The main target lies in the accurate analysis and practical application of one or more of the outlined attacks on a given target. Furthermore, it should be investigated which parts of a QR code are the easiest to attack, and what countermeasures can be taken to thwart attacks like the ones proposed in this paper. In even more general terms, it would be very interesting to find metrics that can be used to measure the vulnerability of QR

codes depending on a given type of attack outline and with respect to characteristics like black/white-distribution, version, masking, etc. In addition, other 2D-Codes such as Aztec [14] or DataMatrix [15] need to be analyzed in the same way to identify possible attack vectors and find suitable countermeasures.

We further want to expand our proof-of-concept attack to consider multiple masks and to take advantage of the balancing effects of the masking itself, as well as switching to different encoding-levels. Especially encoding to different masks could lead to a black-white ratio in our favor. Additionally it would be interesting to determine best-values for the number n of messages
There is also some room for performance optimizations:

- Defining knock-out criterions for the Q_is based on the hamming distance to Q_0, to speed up the validation process.
- Refining these criterions to be applicable to the actual messages M_i to reduce unnecessary code-generation.
- Using a genetic approach for constructing the messages M_i.

Combining all the proposed optimizations, would allow us to perform the attack on devices with limited memory and processing power such as smartphones.

2.7 Conclusion

In this paper we outlined the dangers of possible attacks utilizing manipulated QR codes. Since QR codes gain increasing popularity through their use for marketing purposes, we expect that this kind of attack will receive more and more attention by the hacking community in the future. Furthermore, many mobile devices (e.g., smartphones) are able to decode QR codes and access the URLs contained in them. This adds a new dimension to the topic of trust, especially since most users are not security-conscious enough when using their mobile phones (which also enables the use of novel phishing techniques). We introduced a proof-of-concept phishing attack on QR codes, that is based on the idea of changing the encoded data of a QR code by just turning white modules into black ones. We proposed an algorithm for finding similar QR codes for the attack and showed its feasibility with the help of an example.

In addition to phishing, a multitude of other attack methods, both against humans and automated systems, might be performed using QR codes. This especially holds true if proper input sanitization is not performed prior to processing the contained data.

Acknowledgements

This research was funded by COMET K1, FFG – Austrian Research Promotion Agency.

Bibliography

[1] M. Canadi, W. Höpken, and M. Fuchs. Application of qr codes in online travel distribution. In *ENTER*, pp. 137–148, (2010).
[2] H. S. Al-Khalifa. Utilizing qr code and mobile phones for blinds and visually impaired people. In *ICCHP*, pp. 1065–1069, (2008).
[3] A. Alapetite, Dynamic 2d-barcodes for multi-device web session migration including mobile phones, *Personal and Ubiquitous Computing*. **14**(1), 45–52, (2010).
[4] S. Lisa and G. Piersantelli. Use of 2d barcode to access multimedia content and the web from a mobile handset. In *GLOBECOM*, pp. 5594–5596, (2008).
[5] Y.-P. Huang, Y.-T. Chang, and F. E. Sandnes, Ubiquitous information transfer across different platforms by qr codes, *J. Mobile Multimedia*. **6**(1), 3–14, (2010).
[6] ISO 18004:2006, *QR Code bar code symbology specification*. (ISO, Geneva, Switzerland.
[7] J. Gao, V. Kulkarni, H. Ranavat, L. Chang, and H. Mei. A 2d barcode-based mobile payment system. In *MUE*, pp. 320–329, (2009).
[8] J. Z. Gao, L. Prakash, and R. Jagatesan. Understanding 2d-barcode technology and applications in m-commerce - design and implementation of a 2d barcode processing solution. In *COMPSAC (2)*, pp. 49–56, (2007).
[9] J. Z. Gao, H. Veeraragavathatham, S. Savanur, and J. Xia. A 2d-barcode based mobile advertising solution. In *SEKE*, pp. 466–472, (2009).
[10] I. Reed and G. Solomon, Polynomial codes over certain finite fields, *Journal of the Society for Industrial and Applied Mathematics*. **8**(2), 300–304, (1960).
[11] R. Bose and D. Ray-Chaudhuri, On a class of error correcting binary group codes*, *Information and control*. **3**(1), 68–79, (1960).
[12] A. Hocquenghem, Codes correcteurs d'erreurs, *Chiffres*. **2**(147-156), 4, (1959).
[13] M. R. Rieback, B. Crispo, and A. S. Tanenbaum. Is your cat infected with a computer virus? In *PERCOM '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, pp. 169–179, Washington, DC, USA, (2006). IEEE Computer Society.
[14] ISO 24778:2008, *Aztec Code bar code symbology specification*. (ISO, Geneva, Switzerland.
[15] ISO 16022:2006, *Data Matrix bar code symbology specification*. (ISO, Geneva, Switzerland.

http://www.springer.com/978-94-91216-70-1

Trustworthy Ubiquitous Computing
Khalil, I.; Mantoro, T. (Eds.)
2012, XX, 268 p., Hardcover
ISBN: 978-94-91216-70-1
A product of Atlantis Press