QR Inception: Barcode-in-Barcode Attacks

Adrian Dabrowski
SBA Research
Vienna, Austria
adabrowski@sba-research.org

Katharina Krombholz
SBA Research
Vienna, Austria
kkrombholz@sba-research.org

Johanna Ullrich
SBA Research
Vienna, Austria
jullrich@sba-research.org

Edgar R. Weippl
SBA Research
Vienna, Austria
eweippl@sba-research.org

ABSTRACT

2D barcodes offer many benefits compared to 1D barcodes, such as high information density and robustness. Before their introduction to the mobile phone ecosystem, they have been widely used in specific applications, such as logistics or ticketing. However, there are multiple competing standards with different benefits and drawbacks. Therefore, reader applications as well as dedicated devices have to support multiple standards.

In this paper, we present novel attacks based on deliberately caused ambiguities when especially crafted barcodes conform to multiple standards. Implementation details decide which standard the decoder locks on. This way, two users scanning the same barcode with different phones or apps will receive different content. This potentially opens way for multiple problems related to security. We describe how embedding one barcode symbology into another can be used to perform phishing attacks as well as targeted exploits. In addition, we evaluate the extent to which popular 2D barcode reader applications on smartphones are susceptible to these *barcode-in-barcode* attacks. We furthermore discuss mitigation techniques against this type of attack.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce—*Security*; K.6.m [**Management of Computing and Information Systems**]: Miscellaneous—*Security*

General Terms

security

Keywords

security; protocol decoding ambiguity; barcode; QR; steganography; Packet-in-Packet

SPSM'14, November 7, 2014, Scottsdale, Arizona, USA.

ACM 978-1-4503-3155-5/14/11 ...$15.00.
http://dx.doi.org/10.1145/2666620.2666624.

1. INTRODUCTION

Linear barcodes or *1D codes* are used to provide a machine-readable form of printed information. In cases where higher data density is required, *matrix* or *2D barcodes* are preferentially deployed. Such codes are used in industrial applications, e.g. logistics or tracking of individual components during the production process. In everyday life, electronic tickets are issued with 2D barcodes, and web links are transmitted via 2D barcodes on billboards and in printed ads. Additionally, they are used in security-sensitive applications such as monetary transactions: Paypal and Bitcoin allow shoppers to pay for goods and services using apps that generate QR codes readable by merchants' existing scanning devices [29].

With barcode-in-barcode attacks, two different barcodes are encoded in the same rectangular area, optically appearing as one barcode. This can be accomplished by generating one complex barcode that confirms to multiple standards, or by hiding a smaller one within a bigger one. The latter is similar to Packet-in-Packet attacks known from radio systems [12]. However, this attack relies less on probabilistic errors, and more upon implementation differences of the decoders. It bears resemblance to protocol decoding mismatches, where e.g. a firewall or anti-virus scanner decodes and interprets data differently than the server behind it - allowing an exploit to pass. Any ambiguity is therefore a potential security risk [8, 15, 24].

2. MOTIVATION

Different ways of using QR codes as an attack vector have been proposed [10, 17, 18]. In 2012, hackers showed that *Unstructured Supplementary Service Data* (USSD) codes encoded in 2D barcodes can be used to wipe a phone or execute other system functions [20]. On some phones, they can be used to generate premium rate SMS messages. QR codes can also be used to trigger vulnerabilities in the reader software, the operating system, or a remote site, such as SQL injections [18]. Peng et al. [23] found code injection vulnerabilities in several QR libraries. QR Codes are also used to spread malware [10] and for phishing attacks.

In a variety of attacks differentiation between various phones and bar code readers provides a subtle way of separating one user group from another. The following examples describe respective scenarios for barcode-in-barcode attacks:

(1) It is possible to specifically tailor exploits for a certain

platform or reader application, e.g. the code injection vulnerabilities mentioned above. An alternative would be that one type of phone is wiped, while the others are directed somewhere else to decrease suspicion. Imagine two friends, one trying to decode a bar code with an exploitable phone. As he does not get the expected response from his phone, his friend might try to decode. Due to having another phone, he is directed to a benign homepage. This will avert suspicion for wiping the first's phone from the code.

(2) A malicious graphics designer produces a barcode-in-barcode for a donation campaign which diverts a small percentage of donors to a second account. The siphoned-off money is likely to go unnoticed and the attack is hard to reproduce.

(3) In a less hostile scenario, the ability mentioned above is used to violate users' privacy. Different URLs may expose a user's software or operating system choice to third party web sites. For example, users of top-of-the-market phones (e.g. iPhone) get shown more pricey products than low-end phones. Additionally, this technique can also be used as steganography tool to hide messages in benign 2D codes.

(4) In logistics, a barcode that decodes to different values at different handling points can be used to cheat on fees or cause re-routing and circular routing of parcels.

3. BACKGROUND

There are various *2D* or *matrix barcode* symbologies. Each of them tends to be dominant in one or more particular fields of application. This makes it necessary for many devices to support more than one standard.

Quick Response (QR).

Denso Wave [11] developed the QR code in 1994 and later published the design as ISO/IEC 18004 [2]. Its main features are high data density, robustness, and the ability to encode numbers, text, 8-bit data, and Kanji. The three prominent rectangular finder markers easily distinguish this code and facilitate the correction perspective distortions. The QR code standard features extensive forward error correction (FEC) based on Reed-Solomon codes, allowing large portions of the code to be destroyed without any effects on its decodability. It fits up to 4296 characters or 2953 bytes. Today, this symbology is the first choice when conveying URLs from billboards to mobile phones. Further, it is used in e-ticketing, payment (e.g. Paypal and Bitcon), and (encrypted) visa applications.

Aztec Code.

The Aztec barcode was invented in 1995 and later published as ISO/IEC 24778 [14]. The finder pattern and characteristic visual feature is a quadratic bullseye-style center mark. In contrast to most other codes, by design, it does not require any *quiet zone* around the code (i.e. a surrounding blank region). Data is put in spiral form around the center mark and fits up to 3832 digits, 3067 letters, or 1914 bytes. It also employs Read-Solomon codes for error correction. Aztec barcodes are popular on the transport tickets of several major European rail companies.

Data Matrix.

This code was developed by *ID Matrix* which was later acquired by Siemens [1]. It lacks a primary visual feature as it is surrounded by a thin border on two sides and a dotted line (timing pattern) on the other two sides. This code exists in multiple versions with different error correction algorithms. In 2003, the *Semacode* project started to promote Datamatrix-encoded URLs as standard for the Web. Up until today, Semacode is often used as a synonym. It fits up to 1556 bytes but is very compact on short messages. Therefore, it is used in industrial production to mark (small) electronic components, and in the food industry. It is also used in some countries as digital stamp.

Codes with non-square pixels.

Codes with other forms of picture elements (pixels) are not considered in this paper, since they do not integrate well into square pixel codes. *PDF417* and *micro-PDF417* have been developed for stacked line scanning and use tall and narrow pixels. *MaxiCode* by UPS uses round and octagon elements, and *Microsoft Tag* uses triangles.

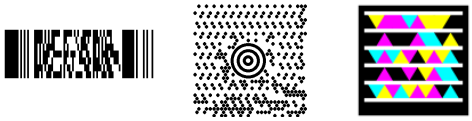

Figure 1: PDF417, MaxiCode, Microsoft Tag

4. INCEPTION PRINCIPLES

"*The nice thing about standards is that you have so many to choose from.*" – Andrew S. Tanenbaum

4.1 Type 1: Multiple Standards Ambiguity

Many barcode readers implement more than one symbology. This can lead to ambiguous situations when a device is confronted with multiple barcodes in one image. A crafted 2D barcode that conforms to multiple standards (or an embedded barcode inside another) would be undetectable to an untrained human viewer. However, decoding software usually employs multiple computationally cheap finders for specific symbologies, e.g. a detector for a specific visual marker of a symbology (Figure 2). When one is found,

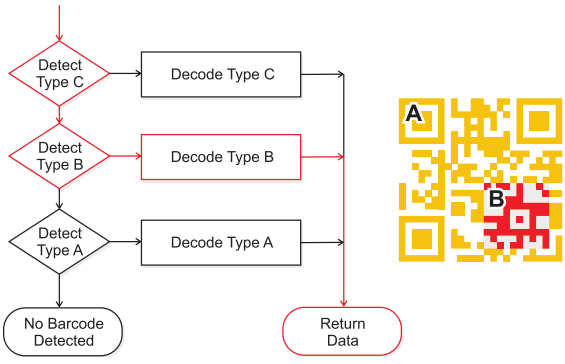

Figure 2: Type 1 Attack: The inner barcode type is always detected before the outer one

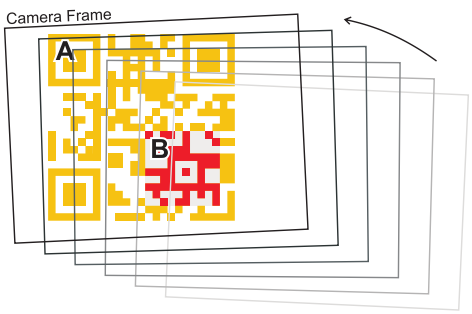

Figure 3: Sliding over the barcode will make the smaller inner barcode fully visible before the entire (outer) barcode

an appropriate decoder retrieves the data and presents it to the user or the calling application.

In this case, we are exploiting the different detection order of reader implementations.

4.2 Type 2: Camera Frame

When the reader application (or mode) starts on the phone, users are unlikely to have the phone pointed exactly at the barcode. While the user is trying to aim for the barcode, they slide and rotate the camera frame (Figure 3). Programs that work on the live imagery and do not require to push a button try to decode every one of these frames. The situation regarding dedicated handheld readers is very similar.

This makes it highly probable, that an (*inner*) embedded code is inside the imaging frame before the full *outer* (or *host*) barcode. However, different implementations might have additional requirements (e.g. size of quiet zones) that also allow for a further discrimination of users.

4.3 Embedding Criteria

Embedding one code into another requires distinct characteristics of the standards for the outer as well as the inner code.

The outer code has to (a) provide a continuous area of a certain size to shelter the other, and (b) a sufficiently robust data correction[1].

The latter is necessary so that the outer code can still be correctly decoded while the scanner interprets the inner one as error caused by the environment or the camera. The robustness of data correction in combination with the actual data size has an impact on the host's maximum. For example, data correction of 30% allows the use of almost one third of the data area for the inner code. The larger the outer data, the larger the area for the parasite within.

Additionally, the outer and the inner code have to consist of the same pixel type. Thus, various codes using squares can be mixed, but cannot be combined with others using rectangles. Finally, to be successful the combined codes must not appear suspicious to the users.

4.4 Types of Hiding a Barcode

Many barcodes require free space (quiet zones) or special markers to be decoded correctly, although the implementations vary. All encountered implementations require substantially less free space around the symbology than specified by the standard. Some require a thin white border, some of them only around the marker elements. Others do not require any separation at all.

[1]or another way to include alien data

Figure 4: Four types of embedding a code

There are essentially four ways to embed a foreign code into a host barcode (Figure 4). They can share two borders and thus provide partial white space for the inner code. Alternatively, it can be submerged somewhere in the middle of the host barcode.

In both versions, the embedded code might provide its own quiet zone or relinquish it. The former makes the embedded inner code easily visible to the human, but can also help the reader algorithms. Aztec is the only compared symbology in this paper that does not require a quiet zone by design.

4.5 QR Barcode as Host

In this paper, we focus on the three codes with quadratic pixels. They provide a uniform look to the untrained eye, minimizing suspicions a potential victim might have. QR and DataMatrix provide a relative large continuous area to hide other codes. Additionally, QR can define *segments* of different encodings – with the side effect that decoders ignore segments with unknown encodings. This allows to fill these segments with arbitrary pixels, as sometimes used to embed icons within QR barcodes. The enhanced FEC allows to simply paste the embedded code over a part of the host code which remains decodable.

In our tests, QR's error correction worked much better than Data Matrix's, which is probably the consequence of employing multiple FEC standards over the time and not having all readers support each of them.

Therefore, currently, the QR symbology provides the best host platform to embed other codes. As versatile as QR's error correction code is, not all parts are protected equally. Some elements are vital and needed before the FEC bits can even be read or applied. Therefore, the embedded code must not interfere with these elements (Figure 5):

Finder or Location Markers These visually prominent markers (including the quiet zone around them) are

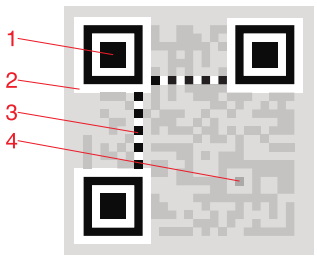

Figure 5: Critical areas of an QR Code: location markers (1), quiet zone (2), timing pattern (3), and alignment markers (4)

Table 1: Barcode standard support and features matrix

OS/Type	Name	QR	Data Matrix	Aztec	Auto-load URLs	Show decoded barcode
iPhone	NeoReader [21]	✓	✓	✓	✓	✗
	Qrafter [16]	✓	✓	✓	✗	✗
	i-nigma [4]	✓	✓	✗	✗	✗
	QR Code Reader and Scanner [27]	✓	✓	✓	✓	✗
	ScanLife [25]	✓	✓	✗	✗	✗
Android	ZXing Barcode Reader [31]	✓	✓	✗	(✗)[1]	✓
	UberScanner [30]	✓	✓	✓	✗	(✓)[2]
	ScanLife [26]	✓	✓	✗	✓	✗
	i-nigma [5]	✓	✓	✗	✗	✗
	AT&T Code Scanner [9]	✓	✓	✗	✓	✗
	NeoReader [22]	✓	✓	✓	✗	✗
	ShopSavvy [28]	✓	✓	✗	✓	✗
Handheld	Symbol DS6708 [13]	✓	✓	✓	–	–

[1] Retrieves URL in background to extract page title [2] Picture excerpt without bounding box

used by the detector to locate a barcode in an image and correct possible distortions.

Quiet Zone The QR standard defines a large white space around each barcode. Most readers still require at least 1 pixel white border around the location pattern, whereas some also manage without a quiet zone.

Timing Patterns These dotted patterns run horizontally and vertically between the inner corners of the three location markers. They are used to synchronize rows and column pixels and are essential for most readers.

Alignment Markers They are only built into bigger codes to help handling distortions. They are less important for most decoders and a limited number of them can be destroyed without reducing readability.

Additional meta data, such as encoding, type of barcode, and level of forward error correction is redundantly encoded around the markers and therefore much more robust.

5. EVALUATION

For the evaluation of our attack we selected the most popular barcode reader applications in the Google Play Store as well as iTunes App Store. The apps were chosen based on popularity and whether they support multiple 2D barcode standards (Table 1). We tested combined barcodes with 5 applications for iPhone and 7 for Android which were all available for free. The goal was to test the applicability, not to provide a market analysis. Furthermore, we tested a professional handheld device as used in retail and logistics.

The chosen handheld decoder was a Motorola/Symbol DS6708 scanner. *Symbol Technologies* (acquired by Motorola in 2007) is the long term leader in handheld scanner devices and predates the mobile phone ecosystem by decades. The device has been reconfigured to enable all supported symbologies.

Measurements have been conducted with an iPhone 4, an iPhone 5, a Galaxy Nexus and an LG Nexus 4 Android phone, and the handheld scanner under normal office illumination. For each barcode we made at least 10 scanning attempts – more if the results weren't conclusive. We varied the distance and rotation between the printed barcode and the sensor.

We refrain from giving percentage numbers of decodes, as exact reproduction depends on multiple parameters, such as illumination, angle, movement, distance and so on. Instead we documented if and how each barcode decoded, and if there was a major preference for one or another.

While iPhone readers display a large variation of behaviors, a large portion of Android apps use the free *Zebra Crossing* (ZXing) [3] barcode library. Therefore, Android apps show a much more uniform behavior. Differences are minimal and are caused by different versions of the used library included. In this case, the *ZXing Barcode Reader* (the demo app for the open source ZXing library) does not support Aztec codes. However, *UberScanner* uses a (newer) beta version of the same library and does supports them.

5.1 Aztec in QR

Aztec is a very good choice for being embedded into another code. By standard it does not require a quiet zone. However, our tests have shown that corner placement (and therefore offering a partial quiet zone) provides a higher decodability rate with the Symbol device.

Qrafter was neither able to decode the inner nor the outer barcode, while NeoReader strongly prefers the inner Aztec code. This is probably a case where the Aztec finder is called before the QR finder. Although tempting, it is not possible

App/Device	Outer	Inner
NeoReader	✓	✓pref.
Qrafter	✗	✗
i-nigma	✓	–
QR Code R.S.	✓	✗
ScanLife	✓	–
ZXing B.S.	✓	–
UberScanner	✓	✓
ScanLife	✓	–
i-nigma	✓	–
AT&T Code S.	✓	–
NeoReader	✓	✓
ShopSavvy	✓	–
DS6708	✓	✓

Figure 6: Aztec in QR: NeoReader on iOS strongly prefers Aztec over QR

to use both finder patterns to overlay each other. The Aztec finder pattern is always centric and a bit taller than QR's. In our tests, QR readers could not be made to accept an Aztec finder pattern as one of the three QR finder patterns or vice versa.

Orientation and placement of the pattern did not play a significant role in decodability.

5.2 Data Matrix in QR

The weakness of Data Matrix is the lack of a distinct visual marker. On the one hand, this makes the code very compact, on the other hand the decoder gets fewer visual clues.

In the first two experiments, we hid the Data Matrix in the bottom right corner of the host QR code without any white space around them (Figure 7). Most readers did not detect the embedded inner code. For NeoReader, the orientation of the Data Matrix code proved to be of importance: a version of the Data Matrix code with the solid line facing the QR's outer border was detected (Figure 8). Interestingly, i-nigma had problems reading the outer QR code: the iOS version did not decode at all and the Android version decoded only after numerous attempts.

App/Device	Outer	Inner
NeoReader	✓	✗
Qrafter	✓	✗
i-nigma	✓	✗
QR Code R.S.	✓	✗
ScanLife	✓	✗
ZXing B.S.	✓	✗
UberScanner	✓	✗
ScanLife	✓	✗
i-nigma	✓	✗
AT&T Code S.	✓	✗
NeoReader	✓	✗
ShopSavvy	✓	✗
DS6708	✓	✗

Figure 7: Data Matrix (bottom right) in QR

App/Device	Outer	Inner
NeoReader	✓	✓
Qrafter	✓	✗
i-nigma	✓	✗
QR Code R.S.	✓	✗
ScanLife	✓	✗
ZXing B.S.	✓	✗
UberScanner	✓	✗
ScanLife	✓	✗
i-nigma	(✓)	✗
AT&T Code S.	✓	✗
NeoReader	✓	✓
ShopSavvy	✓	✗
DS6708	✓	✗

Figure 8: Data Matrix (bottom right, rotated) in QR

In a second test, the embedded Data Matrix code was positioned in the center of the QR code: first without a white border (Figure 9), and then with a white border (Figure 10). The former was not detected by any reader. The latter was decoded by almost all scanners, whereas NeoReader on iOS completely ignores the outer QR code. On Android, ScanLife and the AT&T Scanner only decoded the inner Data Matrix when panning over the image.

App/Device	Outer	Inner
NeoReader	✓	✗
Qrafter	✓	✗
i-nigma	✓	✗
QR Code R.S.	✓	✗
ScanLife	✓	✗
ZXing B.S.	✓	✗
UberScanner	✓	✗
ScanLife	✓	✗
i-nigma	✓	✗
AT&T Code S.	✓	✗
NeoReader	✓	✗
ShopSavvy	✓	✗
DS6708	✓	✗

Figure 9: Data Matrix (center) in QR

App/Device	Outer	Inner
NeoReader	✗	✓
Qrafter	✓	✓
i-nigma	✓	✓
QR Code R.S.	✓	✗
ScanLife	✓pref.	✓
ZXing B.S.	✓	✓
UberScanner	✓	✓
ScanLife	✓	(✓swipe)
i-nigma	✓	✓
AT&T Code S.	✓	(✓swipe)
NeoReader	✓	✓
ShopSavvy	✓	✓
DS6708	✓	✓

Figure 10: Data Matrix (center, white space added) in QR

5.3 QR in QR

QR in QR is a special case. The finder markers compete against each other and may strongly confuse the detector. Additionally, it is easier to be noticed by a human. In this case, the camera frame and angle of rotation can be significant for the software's decoding decision. The results also indicate that some finder pattern algorithms require a white space around the marker, and some do not. However,

App/Device	Outer	Inner
NeoReader	✓	✗
Qrafter	✗	✗
i-nigma	✓	✓
QR Code R.S.	✗	✗
ScanLife	(✓rot.)	✓
ZXing B.S.	✗	(✓swipe)
UberScanner	✗	(✓swipe)
ScanLife	✗	✗
i-nigma	✓	✗
AT&T Code S.	✗	✗
NeoReader	✓	✗
ShopSavvy	(✓)	✗
DS6708	✓	✓pref.

Figure 11: QR in QR, corner, w/o white space

the white space around the whole barcode as defined in [2] is not a necessity for any of the tested readers.

For this series of tests, we increased the level of FEC for the outer barcode, as the inner barcode consumes considerably more area than Aztec or Data Matrix with the same content.

In the first case, as depicted in Figure 11 (QR put in a corner, without additional white space) a significant number of readers had problems decoding any of the QR codes. ShopSavvy decoded very rarely. For the inner code, most of the ZXing based software picked it up panning slowly over the barcode in the moment when the top left corner of the camera frame is aligned with the barcode corner. The DS6708 strongly preferred the embedded code, while i-nigma on iOS was indifferent. ScanLife on iOS picked up the outer barcode only after a significant rotation.

In the case where the embedded code is not exactly in the corner, the recognition matrix changes slightly (Figure 12). i-nigma decodes the inner code slightly better when rotated 45° and slide into the image, while the alignment trick does not work for ZXing-based readers. On Android, i-nigma only decodes the inner QR when it is embedded in the center.

App/Device	Outer	Inner
NeoReader	✓	✗
Qrafter	✓	✗
i-nigma	✓	✓
QR Code R.S.	✓	✗
ScanLife	✓	✗
ZXing B.S.	✓	✗
UberScanner	✓	✗
ScanLife	✓	✗
i-nigma	✓	✓cent.
AT&T Code S.	✓	✗
NeoReader	✓	✗
ShopSavvy	✓	✗
DS6708	✓	✓

Figure 12: QR in QR, semi corner and center, w/o white space

Adding white space around the finder markers of the embedded code (Figure 13) increases the readability dramatically, practically disabling the outer code for many applications. Presumably, implementations prefer markers in close vicinity to each other.

App/Device	Outer	Inner
NeoReader	✗	✓
Qrafter	✗	✓
i-nigma	✓	✓
QR Code R.S.	✗	(✓)
ScanLife	✗	✓
ZXing B.S.	✗	✓
UberScanner	✗	✓
ScanLife	✗	✓
i-nigma	✓	✓
AT&T Code S.	✗	✓
NeoReader	✗	✓
ShopSavvy	✗	(✓)
DS6708	✓pref.	✓

Figure 13: QR in QR, center with white space

Qrafter and ShopSavvy need noticeably longer for decoding, but do so only for the embedded code. i-nigma on Android prefers the outer code when the phone is held further away, and the inner code when held closer to the barcode. QR Code Reader and Scanner on iOS has major troubles with decoding. In our tests it eventually returned the inner code and in one case returned a garbage string.

6. DISCUSSION

The same scanner applications on different platforms can have major differences (e.g. ShopSavvy on iOS exclusively supports QR as 2D symbology) as well as subtle differences (e.g. i-nigma and NeoReader). In general, the results show that it is feasible to select different audiences to decode different content. For example, should another code injection vulnerability be found in a specific library [23], it can be specifically targeted without raising other users' suspicion. Many applications automatically retrieve URLs and display them to the user, exposing the user to phishing (users cannot verify the URL) and to attacks against the html renderer or the operating system such as local buffer overflows (Table 1). The XZing Barcode Scanner loads an external website, but does only display the title to the user. This is sufficient to trigger remote attacks, such as a SQL injection.

None of our tested applications (Section 5) present the user with more than one result at once, if there were multiple codes in one image. Only two applications presented the user with a thumbnail of the barcode and only XZing Barcode Scanner gives the user the exact bounding box of the decoded symbology. Only the latter is sufficient to enable the user to detect these barcode-in-barcode attacks.

6.1 Countermeasures

Countermeasures are distinguished into technical and user-centered approaches. Technical mitigation strategies aim at tackling the issue through methods, like implementation or code standards.

Less robust data correction

Hiding one code in another usually requires robust data correction. Using QR codes as an example, four levels of data correction between 7 % and 30 % are defined. The higher robustness levels were primarily intended for industrial use including a dirty environment or fast movement on a conveyor, but are not necessary for the web-based daily-life use. Thus, the restriction to low levels of robustness in combination with an appropriate maximum size of data would prevent hidden codes while not negatively impacting the code's practicability.

Stringent priority

While the code formats themselves have been standardized, the order of detection is not, but chosen by the software designers. As this is the root cause for code ambiguity, a stringent prioritization should be defined in order to determine which formats should be decoded in favor of others. While this does not prevent hiding one code in another, it guarantees that every reader provides the same content. In the same way, a prioritization of the outer or inner code can be achieved.

As an alternative, a number of measures to raise user

awareness are feasible. They do not mitigate the attack vector itself, but lay a foundation for an informed user decision.

Scanned photo excerpt

Barcode readers can easily present the decoded image and highlight the area of interest containing the decoded barcode. Although a simple-to-implement method, only a minority of readers use such a technique: *ZXing Barcode Scanner* shows a thumbnail image including green markers visualizing the bounding box. *UberScanner* also provides the image but no further detail on the decoded area.

Notification on all codes found

Barcode readers detecting the presence of code ambiguity should present all of them to the user and let her choose the desired one. This requires that software does not stop after the detection of the first code. As our evaluation has shown, scanners lock onto one symbology and never provide the user with multiple contents or formats.

Alien data

Some symbologies (e.g. QR) allow the definition of multiple segments of data in different encoding. As standard, readers will ignore unsupported encodings and skip to the next segment. These segments can be used to hide pictograms and small icons but also other barcodes. A reader application should inform the user of such alien data and warn them about the potential for abuse.

Good QR practice

Ordinary URLs encoded in barcodes hold risks, e.g. containing code to reset the phones, and cannot be identified by the user at first sight. Thus, barcode reader best practice proposes presenting the scanned code or URL to the user and requesting their acknowledgement to access it. Additional URL checks, as proposed in [19] should be conducted.

Nevertheless, this also includes imperatives to the barcode-issuing entity to provide a readable URL that a user can associate an organization, e.g. the official domain name of the organization. URL shortening services or tracking URLs remove the user's ability to visually check the URL.

7. CONCLUSION

In this paper, we presented barcode-in-barcode attacks. We demonstrated that for users with different apps or devices different data is returned when the same barcode was scanned. This is due to deliberately constructed ambiguous barcodes that conform to multiple standards or contain multiple versions of the same symbology within the area of another barcode. We have shown that implementation details cause barcode reader applications to react substantially differently. This remotely resembles Packet-in-Packet attacks on radio devices and protocol decoding mismatch in network protocols [12]. In a similar way, Jana et al. [15] and Alvarez et al. [8] shown how to abuse file-type fingerprinting and parsing differences of anti-virus tools to evade detection. Albertini [6] created perfect *binary polyglots* that are for example valid PDF, JPEG, and ZIP files at once [7]. In general ambiguity is not desired, however in special cases it can be particularly harmful.

The barcode-in-barcode attack allows to discriminate users by their handset (e.g. redirecting iPhone users to more expensive products than Android users) or to precisely target specific platforms with appropriate exploits. In the past, multiple attacks have been presented that used 2D barcodes to inject SQL statements or code into the decoding library, trigger premium rate messages, or call system functions such as a factory reset. Additionally, they can be used for phishing attacks or to spread malware.

As 2D barcodes are gaining importance in security sensitive applications (e.g. financial transactions) they provide an increasingly valuable attack vector for targeted attacks on selected user groups. In Section 2 we presented several attack scenarios based on the applications the barcodes are used in, such as diverting money, exploiting phones, and cheating on fees.

Subsequently, we evaluated the possibilities of this attack with 12 popular barcode applications on iOS and Android as well as a professional barcode scanner as used in logistics. Finally, we proposed countermeasures and mitigation strategies.

Acknowledgments

Part of this work arose during an internship at the National Institute of Informatics, Tokyo. This research was partially funded by the COMET K1 program by FFG (Austrian Research Funding Agency) and the Austrian Science Fund (FWF): P 26289-N23. Moreover this work has been carried out within the scope of u'smile, the Josef Ressel Center for user-friendly secure mobile environments.

8. REFERENCES

[1] ISO/IEC 16022: Information technology – Automatic identification and data capture techniques – Data Matrix bar code symbology specification.

[2] ISO/IEC 18004: Information technology – Automatic identification and data capture techniques – QR Code 2005 bar code symbology specification.

[3] Official ZXing ("Zebra Crossing") project home. `https://github.com/zxing/zxing`, accessed July 18th 2014.

[4] 3GVision. i-nigma. Apple App Store. `https://itunes.apple.com/en/app/id388923203`.

[5] 3GVision. i-nigma Barcode Scanner. Google Play Store. `https://play.google.com/store/apps/details?id=com.threegvision.products.inigma.Android`.

[6] A. Albertini. corkami: Reverse engineering and visual documentations. `http://code.google.com/p/corkami/#Binary_files`, accessed September 6th 2014.

[7] A. Albertini. This PDF is a JPEG; or, This Proof of Concept is a Picture of Cats. In *PoC || GTFO 0x03*. March 2014. `http://corkami.googlecode.com/svn/trunk/doc/pocorgtfo/pocorgtfo03.pdf`.

[8] S. Alvarez and T. Zoller. The death of AV defense in depth? - revisiting anti-virus software, 2008. `http://cansecwest.com/csw08/csw08-alvarez.pdf`.

[9] AT&T Services Inc. AT&T Code Scanner: QR,UPC & DM. Google Play Store. `https://play.google.com/store/apps/details?id=com.mtag.att.codescanner`.

[10] M. DeCarlo. AVG: QR code-based malware attacks to rise in 2012, 2012. `http://www.techspot.com/news/47189-avg-qr-code.html`, accessed July 18th 2014.

[11] DENSO WAVE. History of QR Code. http://www.qrcode.com/en/history/, accessed July 13th 2014.

[12] T. Goodspeed, S. Bratus, R. Melgares, R. Shapiro, and R. Speers. Packets in packets: Orson welles' in-band signaling attacks for modern radios. In *Proceedings to WOOT 2011*, pages 54–61, August 2011.

[13] M. Inc. Symbol DS6708 Digital Scanner Product Reference Guide, 2009. http://www.motorolasolutions.com/web/Business/Products/Bar%20Code%20Scanning/Bar%20Code%20Scanners/General%20Purpose%20Scanners/_Documents/static_file/ds6708.pdf.

[14] ISO/IEC 24778: Information technology – Automatic identification and data capture techniques – Aztec Code bar code symbology specification.

[15] S. Jana and V. Shmatikov. Abusing File Processing in Malware Detectors for Fun and Profit. In *Proceedings of the 33rd IEEE Symposium on Security & Privacy*, San Francisco, CA, May 2012.

[16] Kerem Erkan. Qrafter. Apple App Store. https://itunes.apple.com/us/app/id416098700.

[17] A. Kharraz, E. Kirda, W. Robertson, D. Balzarotti, and A. Francillon. Optical Delusions: A Study of Malicious QR Codes in the Wild. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 06 2014.

[18] P. Kieseberg, S. Schrittwieser, M. Leithner, M. Mulazzani, E. Weippl, L. Munroe, and M. Sinha. Malicious Pixels Using QR Codes as Attack Vector. In I. Khalil and T. Mantoro, editors, *Trustworthy Ubiquitous Computing*, volume 6 of *Atlantis Ambient and Pervasive Intelligence*, pages 21–38. Atlantis Press, 2012.

[19] K. Krombholz, P. Frühwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl. QR Code Security: A Survey of Attacks and Challenges for Usable Security. In T. Tryfonas and I. Askoxylakis, editors, *Human Aspects of Information Security, Privacy, and Trust*, volume 8533 of *Lecture Notes in Computer Science*, pages 79–90. Springer International Publishing, 2014.

[20] B. Naik. QR Code: USSD attack, 2012. http://resources.infosecinstitute.com/qr-code-ussd-attack/, accessed July 18th 2014.

[21] NeoMedia Technologies, Inc. NeoReader. Apple App Store. https://itunes.apple.com/us/app/id284973754.

[22] NeoMedia Technologies Inc. NeoReader QR & Barcode Scanner. Google Play Store. https://play.google.com/store/apps/details?id=de.gavitec.android.

[23] K. Peng, H. Sanabria, D. Wu, and C. Zhu. Security Overview of QR Codes. 2014. MIT Student Paper, available online https://courses.csail.mit.edu/6.857/2014/files/12-peng-sanabria-wu-zhu-qr-codes.pdf.

[24] L. Sassaman, M. L. Patterson, S. Bratus, M. E. Locasto, and A. Shubina. Security Applications of Formal Language Theory. In *IEEE Systems Journal, Volume 7, Issue 3*, Sept. 2013.

[25] Scanbuy Inc. ScanLife Barcode & QR Code Reader with Prices, Deals, & Reviews. Apple App Store. https://itunes.apple.com/us/app/scanlife-barcode-reader-qr/id285324287.

[26] Scanbuy Inc. ScanLife QR & Barcode Reader. Google Play Store. https://play.google.com/store/apps/details?id=com.ScanLife.

[27] ShopSavvy Inc. QR Code Reader and Scanner. Apple App Store. https://itunes.apple.com/en/app/qr-code-reader-and-scanner/id388175979.

[28] ShopSavvy Inc. ShopSavvy Barcode Scanner. Google Play Store. https://play.google.com/store/apps/details?id=com.biggu.shopsavvy.

[29] D. Tam. PayPal offers QR codes for retail-store purchases, October 2013. http://www.cnet.com/news/paypal-offers-qr-codes-for-retail-store-purchases/, accessed July 24th 2014.

[30] Ubercoders. UberScanner. Google Play Store. https://play.google.com/store/apps/details?id=org.ubercoders.uberscanner.

[31] ZXing Team. Barcode Scanner. Google Play Store. https://play.google.com/store/apps/details?id=com.google.zxing.client.android.