QR Code Hacking – Detecting Multiple Vulnerabilities in Android Scanning Software

Joe Homan
joehoman3@gmail.com
Former President & COO
Stephenson Technologies Corp

Jennifer Breese
IST and Cybersecurity Analytics and Operations
Pennsylvania State University
jzb545@psu.edu

Abstract

The COVID-19 pandemic created a need that increased the use of Quick Response (QR) codes. The need to minimize contact with items handled by multiple people, for example, drove restaurants to replace menus with QR codes on tabletops. Scanning a QR code with a smartphone camera presents the user with the option to open a website URL displaying something simple as a restaurant menu or to check-in to health care and trusted payment applications. As with other technology solutions, QR codes come with risks. Accessing a website from a smartphone creates the potential for a cybersecurity hack. For this project the team conducted primary research and investigated cyber risks associated with QR codes. We tested multiple QR code scanning software and documented multiple threat vectors present in many of the publicly available QR code scanning software products. Initial research focused on Android smartphone applications.

Keywords: QR code, Internet of Things hacking, cybersecurity risks, Android smartphones, security awareness.

1. INTRODUCTION

A diner sits down at their favorite restaurant for a nice meal. Thanks to the pandemic, the physical menu is replaced by a Quick Response (QR) Code on a sticker at the corner of the table or a pop-up placard. The hungry diner scans the QR code with their smartphone camera and click on the website to open the menu. As they peruse the menu to make their dinner selection, their personal information is downloaded in the background. The diner's entire contact list, calendar schedule, and other data is being transmitted to a remote site. Without knowing it, the unsuspecting consumer has just been hacked.

Through a project funded by the US Air Force Research Laboratory, researchers at Stephenson Technologies Corporation, a non-profit affiliate of Louisiana State University conducted primary research on hacking of various devices and technologies. In part, this research effort aimed to identify potential threats created by IoT devices, inform the manufacturers of the vulnerabilities, and provide awareness to users and consumers of these devices and technologies.

While QR codes have been around for almost 30 years (Pandey, 2008), their ability to transmit data in a contactless and touchless manner has increased their use during the COVID pandemic. Yet, the risks they pose from a cybersecurity standpoint should be seriously considered before use. This may sound like science fiction or a scare tactic to get consumers to avoid using QR codes, but it is a real threat. QR codes are finding

increased use in more industries and media outlets including "commercial tracking systems, entertainment, in-store product labels, marketing, traditional print newspapers, television broadcasting, traditional book publishing and websites" (Akta, 2017, p. xxiii) This paper will present the primary research conducted on QR code vulnerability, the results of analysis of QR code software, identification of specific software applications with inherent vulnerabilities, and recommendations for further research.

Figure 1: 1D Barcode

2. QR CODES IN USE

What is a QR code? Essentially it is like the barcode that is commonly used on grocery items and scanned at the checkout register. Barcodes (Figure 1) code information in only one direction (one dimension), while QR codes store information in 2D, that is in two directions: left/right and up/down (Figure 2). The advantage of 2D QR codes over 1D barcodes is that they can hold much more information. A 1D barcode can hold 85 characters while a 2D barcode "can contain 7,000 digits of characters at maximum including Kanji characters" (Pandey, 2008, p. 60) Barcodes continue to be prevalent in grocer and other applications and 2D codes have become more popular in advertisements (as witnessed on television during the 2022 Super Bowl), automated systems, transportation, and restaurants.

Figure 2: 2D QR Code

Newer QR code or similar 2D code designs have been developed or proposed to increase data storage. For example, "a new high capacity color barcode, named HCC2D (High Capacity Colored 2-Dimensional), which use colors to increase the barcode data density" (Querini et al., 2011, p. 136) has been proposed and the authors considered the hacking and security aspects of such barcodes. Other research into designer QR codes (which may include images like corporate logos) has studied use of color and contrast on the ability of the QR code to be recognized but did not consider hacking or security aspects (Berisso, 2013).

Whether the QR code hacking is related to the release of personally identifiable information (PII) in the healthcare arena (Ang, 2021), in high-tech artificial intelligence applications (Wahsheh & Al-Zahrani, 2021) or something as mundane as opening a menu in a restaurant, the risks are real, and consumers and users should be aware of the dangers associated with QR code use. Depending upon the security features of the software application that reads the QR code, data on the device may be safe – or exposed (Wahsheh & Luccio, 2020).

3. THE PROBLEM

QR codes can be used to hack users' electronic devices, most notably – smartphones and tablets. Both QR codes and smartphones have become ubiquitous, the former especially during the pandemic. Increased usage of any technology that touches personal, commercial, industrial, or military systems creates an opportunity for hackers. Most cybersecurity attacks revolve around hackers attempting to improperly access information (personal or corporate) on individual devices (computers, smartphones, tablets, etc.), i.e., phishing. Previous research theorized about the "dangers of possible attacks utilizing manipulated QR codes" and the expectation "that this kind of attack will receive more and more attention by the hacking community in the future. (Kieseberg et al., 2012, p. 37) These authors outlined the concepts that may lead to hacking but did not directly test various methods or prove the feasibility of hacking a QR code.

The focus of this research was to determine if it is possible to perform more advanced attacks with QR codes in smartphone applications, specifically those with the Android operating system. These attacks included executing malicious commands on a victim's smartphone. Such commands could be used to carry out more advanced attacks (e.g., command injection). Based on our initial research, it appeared there was little research done in this domain. As noted by (Atka, 2017) "[n]ot much literature exists on

QR (Quick Response) Codes and their applications in the emerging digital society." One early study noted "it is not likely that users will be able to find out easily the content encoded, typically URLs, until after they scan QR codes. This makes QR codes a perfect medium for attackers to conceal and launch their attacks based on malicious URLs." (Yao & Shin, 2013, p. 341) Other researchers have suggested that QR codes can be used as an attack vector for SQL injection or command injection. (Focardi, 2019) Previous research documented attacks as largely theoretical, experimental, or simulated rather than actually testing the limitations and risks associated with QR codes. (Zhang et al.,2021; Mavroeidis & Nicho, 2017)

Beyond academic research and theoretical suggestions of hacking dangers, our literature review revealed minimal direct testing and awareness surrounding the real risks associated with QR codes. The initial focus of our research was on achieving various types of command injection with QR codes on the Android smartphone platform. Our hypothesis was that QR codes could be used to inject commands into applications that do not perform proper input sanitization to prevent malicious activity. Our project focused on achieving various types of command injection with QR codes on the Android platform

4. RESEARCH APPROACH

QR codes only store data. They are a static image and do not do any processing of the data. Software on the device that scans the QR code reads the data and directs the device to the appropriate website (Pandey, 2008). The software should have appropriate controls in place to prevent phishing attempts and not execute malware that a QR code could point to. If the scanning software application is not secure, it can execute malicious commands packaged inside of the QR code (Focardi, Luccio, & Wahsheh, 2019).

The approach of our research effort was to identify potential methods of hacking using QR codes to execute malicious commands on various devices. We obtained and tested multiple QR code software applications to evaluate them for vulnerabilities to multiple hacking methods. Having the highest market share globally makes Android a target for attacking by exploiting vulnerabilities. Further, due to the complexity and specialization of the vulnerabilities, few users can relate them to their mobile devices (Wu et al., 2015). This research began with Android devices and could be applied to other operating systems including those on Windows and IOS smartphones.

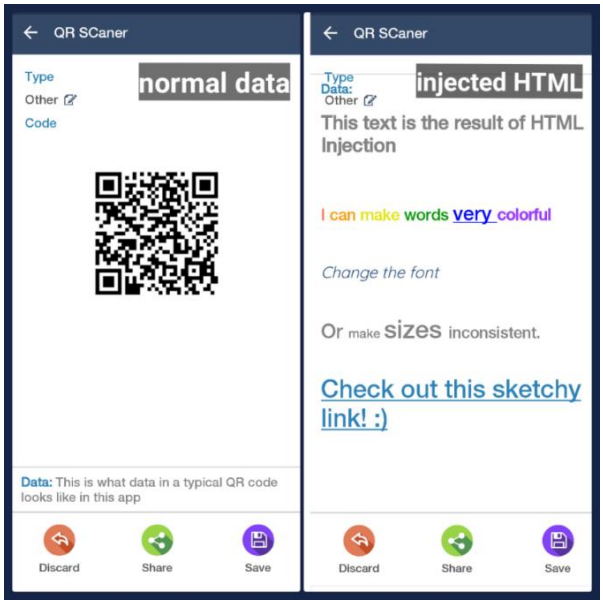

Figure 3: HTML Injection

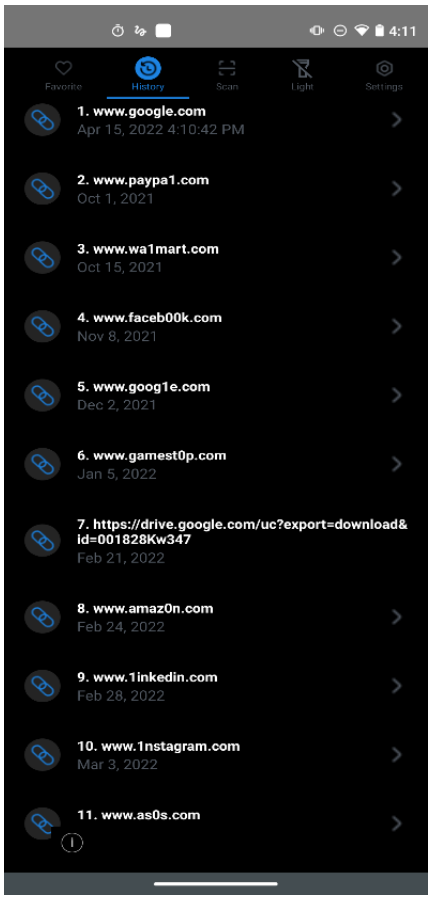

Figure 4: SQL Injection

5. METHODOLOGY

Multiple categories of potential hacks and risks were identified to test for potential security issues. These attack vectors and a brief description of each is as follows:

- HTML injection: In this hack, the application gets tricked into parsing html within the QR code, causing unintended appearance modifications. HTML (HyperText Markup Language) is used for formatting text on websites, so during our testing, we could do things such as change the color, size, and style of text inside of the application. (Figure 3)
- SQL injection: In this attack, the application gets tricked into executing SQL (Structured Query Language) commands on a database. SQL can be used to insert and delete data inside of a database. With this attack, we were able to insert entries into the user's "scan history" database, which is a database that shows them all previous scan results from QR codes they have scanned. By putting an insert statement in the QR code, we were able to insert dozens of malicious websites directly into their scan history. (Figure 4)
- Javascript injection: This hack only works for vulnerable web based QR code scanners, but the app gets tricked into executing JavaScript within the QR code. It is important to note that our research did not find an Android-based device application published in the Google PlayStore that was vulnerable to this type of injection. Web-based scanners are less common but to prove the vulnerability is real, our researchers wrote a demo QR Code scanner that contained this vulnerability. This allowed us to change the HTML using JavaScript, causing drastic appearance changes. Future research will focus on investigating a spyware concept using Javascript injection.
- Unsupported encoding modes: By standard, QR codes only support numeric, alphanumeric, kanji, and byte encoding modes. We created a few methods that allowed other types of characters (such as Unicode) to work inside of QR codes, but the QR code scanner must include support for it. We experimented with putting some of these Unicode symbols inside of the QR code to see if scanning them would bring about unintended behavior. This would allow us to bypass any input checking the QR code software was doing.
- Special symbols: The collection of QR codes that we created didn't target any specific type of injection but included some symbols that had significance across a wide range of programming languages. This is the group of codes we used to test and prove that the above attacks that were possible.
- Research included symbols that have significance in various languages. The basic testing methodology involved the following steps:
- Feed QR codes into a sample of scanning applications and observe behavior
- Determine if behavior was as expected or if abnormal or inappropriate actions took place
- Once suspicious behavior was found, pass the application through reverse engineering and debugging tools
- Analyze logs, decompile Java, and evaluate Java Virtual Machine (JVM) bytecode to pinpoint the issues and potential vulnerabilities
- Craft more QR codes to exploit the vulnerabilities, once identified

The following process was followed in conducting this research:

1. Created a folder filled with QR codes that contained special symbols. These are the same "special symbols" discussed above. They contained characters like ";", "<", " ` ", and others that have significance in programming languages.
2. Scanned these QR codes with a wide range of QR code scanners (both default scanners preinstalled on the phone and ones downloaded from the Google PlayStore) and observed how they responded.
3. Observed that a collection of scanners was not responding to the "<" symbol, and a group of other scanners would crash when the camera scanned the " ` " symbol.
4. Ran the problematic applications through a decompiler, which is a software that tries to reconstruct the original application code. Decompilers work like a compiler, but in the reverse direction. This let us pinpoint exactly where the abnormal behavior was occurring, why, and whether it was due to a vulnerability.
5. It was determined that the applications that weren't responding to "<" had an HTML injection vulnerability. The app was reading the symbol as an HTML tag which has the general format <tag>contents</tag>. The QR code software interpreted the symbol in the QR code as a part of its own app code. The software did not recognize anything inside of the QR code for it to process and therefore did not take the appropriate action.

6. The applications that were crashing as a result of scanning " ` " had a SQL injection vulnerability. The symbol is used inside of SQL statements. The application was carelessly passing the QR code text directly into a SQL statement. When the QR code text contained " ` ", it created a syntax error in the statement and caused the app to crash. As in the HTML injection case, the app was tricked into thinking the symbols we inserted in the QR were a part of its application code and was trying to execute it.
7. As we learned more about the vulnerabilities, we constructed QR codes that exploited these vulnerabilities as demonstrated in Figures 3 and 4.

For the JavaScript injection, we created the demo web-based scanner as a proof of concept. We did so by writing a web application that was susceptible to JavaScript injection, then loading it into an android app we created using WebView. WebView is a technology used by Android application developers to turn an existing website into an application. Many developers are attracted to this since they can create a mobile phone application by recycling website code. Once this was set up, we were able to prove that arbitrary JavaScript could be injected through QR codes if scanners are designed this way. The second and following pages should begin 1.0 inch (2.54 cm) from the top edge. On all pages, the bottom margin should be 1-1/8 inches (2.86 cm) from the bottom edge of the page for 8.5 x 11-inch paper. (Letter-size paper).

6. RESULTS

Based on initial research, there were no observed vulnerabilities in native QR code scanning software (i.e., the applications built into Android devices). However, multiple applications downloaded from the Google PlayStore were susceptible to HTML and SQL injection. Figure 5 graphically depicts the types of QR code software vulnerability types.

Using the methodology described above, the following steps and results were observed:

- Wrote an experimental QR code scanner that executes JavaScript commands
- At least one of the scanners in the Google PlayStore appeared vulnerable to a JavaScript injection
- A small number of apps crashed or gave unexpected behavior
  - For some, due to poorly written code
  - For others, due to injection vulnerabilities
- After more reverse engineering, crafted QR codes to achieve injection
- Types achieved: HTML, SQL, JavaScript
  - No vulnerabilities found in native QR code scanners
  - Some third-party scanners were found to have HTML and SQL injection vulnerabilities
  - These vulnerabilities were a result of not sanitizing input properly and passing it directly to Android application programming interface (API) calls such as from Html and exec SQL.
  - If the application uses an Android Webview, it is susceptible to extensive HTML injection and even Javascript injection, if the webpage it uses is susceptible to these things.
- innerHTML: allows insertion of arbitrary html tags
- JavaScript enabled: allows for JS injection
  - Extensive HTML injection was found in one app on the Google PlayStore. JavaScript injection vulnerabilities weren't found in the wild, but we were able to achieve it experimentally

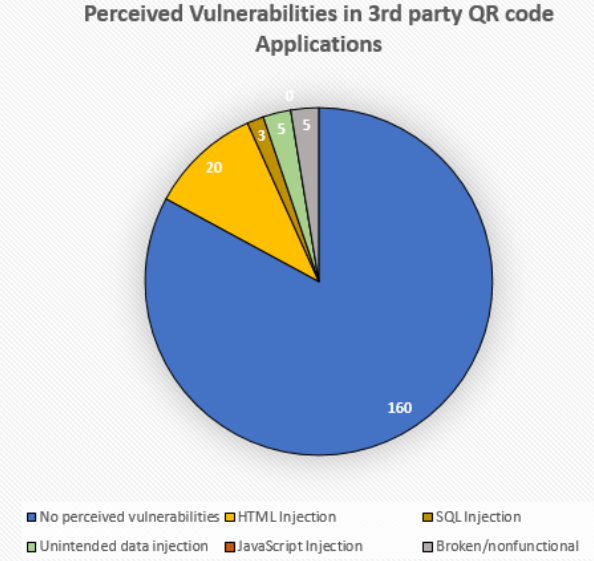

Figure 5: QR Code Software Vulnerability Types

7. RESPONSIBLE DISCLOSURE

Responsible disclosure is a process that cybersecurity researchers typically follow to report vulnerabilities discovered during their research to the appropriate parties. The intent is to allow software manufacturers (in this case) to address vulnerabilities prior to research being published and the risks being disclosed to the

public (and, of course, hackers). Although there is no standard period defined between notification and publication of research, a minimum of 30 days is recommended.

All manufacturers of QR code software that demonstrated a vulnerability in the three categories identified in this paper were notified of one or more vulnerabilities detected in their software. The notifications were sent in February 2022, and all were given the opportunity to contact our researchers for additional details associated with their vulnerability.

Figures 6, 7, and 8 identify companies and their application in the three categories of vulnerability – namely HTML injection, SQL injection, and data injection.

App Name	Company Name
AiScanq	ViewAI Lab
Code Scanner	QR Code Apps
QR & Barcode Scanner	Rajshah5599
QR &Barcode Scanner	BondRen
QR and Barcode Scanner	Apps360 Team
QR and Barcode Scanner	Solution App Technology
QR Code	Faysal INC
QR Code Copy and Save	FalsinSoft
QR Code Scan	ZipoApps
QR Code Scanner	bestdeveloperteam
QR Code Scanner	Hotapp Studio
QR Code Scanner Plus	Disneysoft
QR Scanner	1MB Apps Studio
QR Scanner	App Karo
QR Scanner	TOH Talent Team
QR X2 - Scan QR & Barcode	Easy To Use (OnMobi)
QR/Barcode Scanner Pro	Smart Scanner
Quick-QRScan	Madhivanan

Figure 6: QR Code Software Vulnerable to HTML Injection

App Name	Company Name
QReader3	Dmitry Ventures
QRMagic	Boris Expert
QR Code Scanner	MobMatrix Apps

Figure 7: QR Code Software Vulnerable to SQL Injection

App Name	Company Name
QReader3	Dmitry Ventures
QRMagic	Boris Expert
QR Code Scanner	MobMatrix Apps

Figure 8: QR Code Software Vulnerable to Data Injection

8. CONCLUSION

QR code users may not see a cybersecurity risk from that small square patch of dots. Many who have become accustomed to using them to view restaurant menus do not think twice about continuing to do so. However, the software reading QR codes may pose a bigger threat than users ever imagined. If scanning applications do not have the necessary security protocols in place, the user may be at risk of data loss.
An easy way for a hacker to attack a smartphone would involve a simple switch as follows:

1. The hacker uses the correct QR code to access a restaurant's menu.
2. The menu is copied and duplicated on a different website.
3. A new QR code is created that directs the scanning software to the fake website.
4. The new QR code is pasted over the restaurant's QR code at the table.
5. The diner scans the QR code and sees the menu on the fake website.
6. The malicious site uses HTML, SQL, or data injection to infect the smartphone and download personal data like contacts, calendar events, email, etc.
7. The user orders their meal completely unaware that they have been phished.

It is vital to educate developers about the importance of software development using secure development operations. The SecDevOps process prioritizes delivery speed, integrated security, and system quality. The process is a continuous integration/delivery pipeline that weaves security awareness, practices, and testing through the fabric of the development, testing, and deployment process instead of incorporating it into just one phase like a DevOps process. Software developers must design their code to check inputs to a system to prevent injection of malicious code. This will help strengthen security of current APIs offered to consumers. Equally, consumers need to understand the risks associated with third party applications for software like QR code readers.

This primary research was conducted by engineers, analysts, and a student intern at Stephenson Technologies Corporation in Baton Rouge, LA. The research, partially funded by a US Air Force Laboratory contract, uncovered the vulnerabilities identified in this paper. As noted, we followed the responsible disclosure protocol by informing the manufacturers of the QR code software containing vulnerabilities so that they could address the issues before the publication of this paper.

9. FUTURE WORK

This research was limited to the Android operating system. Future research is needed to repeat this work for Windows and IOS QR code scanners. Also, this research focused on smartphone software applications. Future research is also needed to assess QR code scanners used in other applications such as those used in airports, public transportation, banks, factories, hospitals, warehouses, etc.

10. REFERENCES

Ang, R. J. (2021). Encrypted QR Code in Healthcare Applications. *Canadian Journal of Nursing Informatics*, *16*(2).

Akta, C. (2017). *The Evolution and Emergence of QR Codes* (1st. ed.). Cambridge Scholars Publishing, Newcastle upon Tyne, GBR.

Berisso, K., (2013), Designer QR Codes; Ensuring the "beep," Ohio University Department of Engineering Technology and Management White Paper.

Focardi, R., Luccio, F. L., & Wahsheh, H. A. M. (2019). Usable security for QR code. *Journal of Information Security and Applications*, *48*, 102369. https://doi.org/10.1016/j.jisa.2019.102369

Kieseberg, P., Schrittwieser, S., Leithner, M., Mulazzani, M., Weippl, E., Munroe, L., & Sinha, M. (2012). Malicious pixels using QR codes as attack vector. *Atlantis Ambient and Pervasive Intelligence*, 21–38. https://doi.org/10.2991/978-94-91216-71-8_2

Mavroeidis, V., Nicho, M. (2017). Quick Response Code Secure: A Cryptographically Secure Anti-Phishing Tool for QR Code Attacks. In: Rak, J., Bay, J., Kotenko, I., Popyack, L., Skormin, V., Szczypiorski, K. (eds) Computer Network Security. MMM-ACNS 2017. Lecture Notes in Computer Science, vol 10446. Springer, Cham. https://doi.org/10.1007/978-3-319-65127-9_25

Mohammed S. Al-Zahrani, Heider A. M. Wahsheh, Fawaz W. Alsaade, "Secure Real-Time Artificial Intelligence System Against Malicious QR Code Links," *Security and Communication Networks*, vol. 2021, Article ID 5540670, 11 pages, 2021. https://doi.org/10.1155/2021/5540670

Pandey, D. (2008). Synthesis Journal, Retrieved from https://www.academia.edu/31427962/-Three_QR_Code

Park, H., Kim, T., Kim, G., Jang, W., Lee, K., & Lee, S.-Y. (2019). Improvement of QR code access control system based on Lamport Hash Chain. *Innovative Mobile and Internet Services in Ubiquitous Computing*, 824–833. https://doi.org/10.1007/978-3-030-22263-5_79

Querini, M., Grillo, A., Lentini, A., & Italiano, G. (2021). 2D Color Barcodes for Mobile Phones, *International Journal of Computer Science and Applications*, Technomathematics Research Foundation, Vol. 8 No. 1, pp. 136 – 155.

Wahsheh, H. A. M., & Al-Zahrani, M. (2021). Secure real-time computational intelligence system against malicious QR code links. International Journal of Computers, Communications and Control, 16(3). https://doi.org/10.15837/ijccc.2021.3.4186

Wahsheh, H. A. M., & Luccio, F. L. (2020). Security and Privacy of QR Code Applications: A Comprehensive Study, General Guidelines and Solutions. *Information*, *11*(4), 217. MDPI AG. Retrieved from http://dx.doi.org/10.3390/info11040217

Wu, J., Wu, Y., Yang, M., Wu, Z., Luo, T., & Wang, Y. (2015). QRCloud: Android Vulnerability Query and Push Services Based on QR Code in Cloud Computing. *Journal of Computational Information Systems, 11*(11), 3875–3881.

Yao, Y and Shin, D. (2013). Towards preventing QR code based attacks on android phone using security warnings. In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security (ASIA CCS '13). Association for Computing Machinery, New York, NY, USA, 341–346. https://doi.org/10.1145/2484313.2484357

Zhang, J., Liu, S.-J., Pan, J.-S., & Ji, X.-R. (2018). Digital Certificate based security payment for QR code applications. *Proceedings of the Fourth Euro-China Conference on Intelligent Data Analysis and Applications*, 88–97. https://doi.org/10.1007/978-3-319-68527-4_10