QR Codes for Authentication and Authorisation

Andy Hansen

Supervised by David Eyers

July 17, 2014

Contents

1 Introduction

QR codes have been around since 1997, but have dominantly putting website URLs into a physically scannable form. The aim of my project is to see if there are interesting ways we can use QR codes as a way of authenticating a user, transferring privileges between a user's devices, and setting up a secure channel of communication between a user and the services they wish to access. We plan to use Android smartphones as a means of generating and displaying the user's credentials in QR code form, so that they may be scanned by the service and grant them access without the user having to enter their username and password directly where it could be compromised. The user's details will sometimes be combined with context information proving that the intended user is the one scanning the QR code or codes.

In this report I am going to give an overview of my infrastructure, explain what my system is currently capable of, give the advantages of technologies I have picked, and talk about what I will be doing in the future.

2 Background

2.1 Kerberos

Kerberos [7] is a computer network authentication protocol that allows devices to prove their authorise and authenticate one another in a secure way. It uses tickets as its mechanism to prove identity, a valid user will have a ticket to give to the service they wish to access. Kerberos allows both the user and the server to identify each other. When a user logs into the Kerberos key distribution center (KDC) they are given a ticket granting ticket (TGT). The TGT is presented by the user when they wish to access a restricted service, if the service accepts the user's TGT they will be given a ticket specific to the service when they can then use to access it securely. Kerberos is single sign on meaning that once a user gets their TGT, they will not need to login again until the TGT expires.

2.2 QR Codes

A QR, or Quick Response code [5] is a specially formatted image which is designed to be quickly read by a camera. QR codes come in a variety of 'versions', a version refers to how many rows and columns there are in the code. A high version code is going to be able to store more information, but will also be harder to read. QR codes store data using one of four different modes: numeric only, alphanumeric, byte/binary (ISO8859-1), and kanji. The mode affects how many characters can be stored within the QR code e.g. a numeric only code will be able to store more than the alphanumeric code.

3 Things to Consider

There are many things I need to consider in this project to increase convenience and reduce risk for the user. In this section I discuss the things in my project that are going to need finetuning to if I want the best results.

3.1 Kerberos Ticket Expiry Times

I need to consider the ideal ticket expiry time. The attacker's effective time window needs to be reduced as much as possible without inconveniencing the user. Since the tickets are held on the users phone, I want to make the tickets expire faster but allow the user to easily get more tickets once they do. I will need to do some research to find the ideal amount of time for a ticket to stay alive so that a user does not feel inconvenience, will still making it hard for an attacker to use the ticket for very long after it has been intercepted. Kerberos is able to issue renewable tickets, I can take advantage of this feature and have tickets with short valid times that can be refreshed by the user by recontacting the KDC.

3.2 QR Code Versions

When there are many Kerberos tickets to encode it will take multiple QR codes to store all of the information. It is important that users do not feel inconvenienced by this, so the right QR code version needs to be used. We want a code that is fast to scan and uses a minimal amount codes. Picking the right QR code version is quite difficult because there are many factors to consider:

screen size, pixel density, and camera quality when scanning the QR codes. A ticket can easily fit into a version 40 QR code, but it is very hard for a scanner to read it, especially when the QR code is displayed on a screen with low resolution. Tickets around version 20 can be reliably and quickly scanned, but this does not mean they are the ideal for all smartphones. QR code version will vary depending on the screen resolution. I will be performing some tests in semester 2 to find the ideal code versions so I can use as few QR codes as possible, and still have them easy to scan.

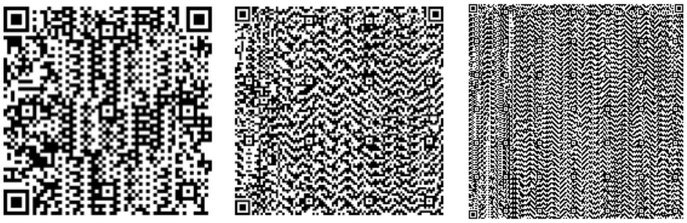

Figure 1: QR code versions 10, 20, and 40 respectively.

4 Project Progress to Date

In this section I give an overview of what I have achieved so far, and the advantages of the technologies I have chosen, and how my project differs from my original plan.

4.1 Infrastructure

My current infrastructure involves two servers that are both running Ubuntu 14.04. The first server is used as the Kerberos key distribution center (KDC) running version 5 and a domain name server that uses Bind version 9 [8]. The second server hosts an Apache web server which can be only be accessed with a valid Kerberos ticket from the KDC server. At the moment the Apache server is playing the role of any possible future Kerberos enabled service. It is a good service to use when testing because there is instant visual feedback when it is working. A use case that results in me being able to access the Apache server could be replaced with any other service such as SSH or access to my account from the user login GUI.

I am using Kerberos because it is runs on all popular operating systems and has built in support in a lot of existing services such as Apache, SSH, and Samba. This means that when a service is configured to connect with the user's KDC, they can use their TGT to access restricted web pages, ssh into protected machines, and access specific folders. I am using Kerberos version 5 because it is the most recent and has resolved some security concerns that were present in version 4 [6]. Kerberos is also useful because all tickets have an expiry time. This is important in my system because if a malicious user intercepts the QR codes then they can only abuse the tickets in the short time window before they expire. Kerberos is single sign on which is useful in our QR code based system because the user can use the TGT to gain access to any new service they need without reentering their username and password. It is a protocol that is supported on all major operating systems and is well tested. If I was to try and design an authentication protocol myself it would require me to do all of the integration and testing which would take a long time without providing very much benefit.

In my project QR codes are going to be used as a primary communication channel between the user and their services. QR codes are a good option because they are easy to use, and hard to perform a man in the middle attack on. The malicious user would have to get a copy of all the user's QR codes before they could use them themselves.

Unfortunately QR codes do not have support direct binary encoding, this means that Kerberos tickets (which are stored in binary) need to be converted to an alphanumeric format before being encoded. I am currently converting the tickets to the string representation of hexadecimal but in the future I will convert them to base64 because it makes more use of the alphanumeric character set, allowing QR code sizes to be reduced further.

4.2 Proof of Concept

I am going to give a run through of my basic use case to show how a users ticket can be transferred from one machine to another using QR codes [4]. Though basic, it proves that it is possible to for a program to be implemented using a phone which can store Kerberos tickets as QR codes, and for a scanned QR code to be put into the ticket cache as a valid Kerberos ticket. My proof of concept operates as outlined here:

- Both computer A and B have no Kerberos tickets, and therefore are unable to access the Apache server.
- Computer A runs `kinit` which is a command line program used to authenticate a user to the Kerberos KDC and get their TGT. They enter their details and are given their TGT. They can then use this ticket to negotiate a ticket for the Apache server, granting them access to its resources.
- Computer A then runs the QR code creating program. The program takes the TGT from the ticket cache (the location Kerberos tickets are stored) and converts it to hex, it then takes the hex and splits it into small sections. Each of those sections are then encoded into a QR code with a number used to identify the order of the QR codes so they can be reassembled.
- Computer B, which is running the QR code scanning software, scans each of the QR codes created by Computer A. When all the codes are decoded they are reassembled using the ordering numbers from before, converted back to binary, and then added to the Kerberos ticket cache. Computer B is now able to use the TGT just as Computer A could before to negotiate a ticket to access the Apache server.
- The tickets from computer A have now been transferred solely using QR codes as the primary means of communication.

In the future this use case would replace computer A with an Android smartphone, they would be able to scan their code on any computer running my QR code scanning program to get authenticate themselves and gain access to the services they are allowed to, such as the Apache server above.

4.3 Difference From Aims and Objectives

I originally planned for there to be more interaction with the phones sensors as a way of confirming the phone is in the same location as the scanned QR code. Instead of this I am now having all of the information coming from the QR codes in most use cases. This puts fewer hardware requirements on the device scanning the QR codes because they do not need to have hardware support something like Bluetooth or near field communication. GPS information is also only available outdoors so will be a logical choice in fewer use cases than I originally intended. We are going to allow users to prove their location by scanning room specific QR codes which I feel is a good trade off between security and ease of use. It is more reliable than GPS when indoors, and does not require extra sensors like Bluetooth or near field communication would. It is possible for the room specific QR code to be copied, but so could the MAC address of the phone if a malicious user was so inclined.

5 Related Work

There is nothing which does what I am trying to achieve so I had a look at projects that use either a mobile phone, or a QR code for similar purposes to my project. I wanted to see where their faults were so I could try and avoid them when implementing my own use cases.

5.1 Web Authentication Using A Mobile Phone

This project allows a user to put a proxy between themselves and an untrusted computer for their websession [9]. The proxy server stores the usernames and passwords. A text message is used to authenticate the user's session to the proxy, and the proxy acquires the login sessions for the user so they do not have to enter their details on the suspicious computer.

My solution takes a different approach to this one, theirs can run on any computer because they just need to connect to their secure proxy. I sacrifice the ability to work anywhere for allowing more uses than just the web, and users of my program can transfer their permissions from one computer to another without having to enter their username and password again.

5.2 QR Code Based Door Access

This project uses QR codes to open doors [1]. The QR code does not expire which is dangerous, but it seems the main purpose of the project is for convenience over security. The idea behind their project is that a user can be emailed their access codes and seem intended to replace key cards. The problem with this method is that replay attacks could be set up to copy a user's QR code and give it to the attacker.

My implementation differs from this because rather than storing an access key, the QR codes in my system store the TGT which could be used to get the user a login session at a computer as well as room access. Their system sends QR codes via email. If someone was able to perform a man in the middle on their mailserver they could gain access to every QR code emailed out.

6 Future Work

A big part of my project is coming up with ways the QR codes can be used and then implementing them. We have come up with some potential use cases, and this semester I am working out the specifics involved in the use cases and implementing them. Im going to go through some of my considered use cases and work out any of the details. I also need to implement the Android application so that these use cases can be carried out in a portable manner.

6.1 Android Application

In my proof of concept I have shown what my system is capable of doing, but in the future I want the user to be able to start the interaction with an Android smartphone. The reason Android is the target platform is because MIT Kerberos has been ported to it [2]. There is also an open source application I can use as a reference on how to build my own Kerberos application [3]. My use cases should have low processing requirements so even modest processors will be able to run my application. Smartphones also come with cameras, Wi-Fi, and displays built in. This means that my use cases can include the user connecting to the internet, scanning QR codes with their camera, or displaying QR codes on their screen without fear that user will not have the hardware to perform these actions.

6.2 Collaboration On a LAN

In the future I want to be able to use the Android application as a means of establishing trust between a group collaborating on a LAN. The master user could create a QR code which can be scanned by all the other users. They use the information in this QR code to set up a trust relationship between their own KDC and the KDC of the LAN they are collaborating on. Once this trust is set up, each of the users can use their own Kerberos account to access services on the network they are collaborating on without having to get their own account for the LAN. It also means that the owner of the LANs KDC can discontinue the trust with any KDCs of users leaving the LAN. With this use case there is potential for a user to cause damage to any services they are give access rights to, but the user who caused the damage will be known because they must use their own account, and the trust can be broken with them at anytime, stopping their access.

6.3 Transferring a Login Session

I also want to allow a user to take their login session elsewhere using their phone. The idea is that if a user has already logged in then they should be able to take that session with them on their phone and continue it on the same network without entering their details again. Upon logout the user has the option to receive QR codes for their phone which contains their TGT. They can scan the QR codes from their phone at any other computer on the network running my QR code scanning program and it will place their TGT into the ticket cache and log them in without them having to enter their details. For this use case to operate correctly I need to be able have my program as part of the login GUI. A possible threat with this use case is a malicious user 'stealing' a login session, when the real user is out of the room. This is something I will need to consider when making the my program.

Bibliography

[1] Jeremy Blum. Libetech qr code door lock. http://www.jeremyblum.com/portfolio/libetech/, 2014.

[2] Chris Conlon. cconlon/cconlon/krb5-anonsvn. https://github.com/cconlon/krb5-anonsvn, 2014.

[3] Chris Conlon. cconlon/kerberos-android-ndk. https://github.com/cconlon/kerberos-android-ndk, 2014.

[4] Andy Hansen. Qr code proof of concept. https://www.youtube.com/watch?v=v8ZZWC-jXeM&list=UU3CfgH3Wtm0TTpxq0I92XFw, 2014.

[5] Denso Wave Incorporated. What is a qr code? http://www.qrcode.com/en/about/, 2014.

[6] MIT Kerberos. Kerberos version 4 end of life announcement. http://web.mit.edu/kerberos/krb4-end-of-life.html, 2014.

[7] B.C. Neuman and T. Ts'o. Kerberos: an authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, Sept 1994.

[8] Douglas Brian Terry, Mark Painter, David W Riggle, and Songian Zhou. *The berkeley internet name domain server*. University of California, 1984.

[9] Min Wu, Simson Garfinkel, and Rob Miller. Secure web authentication with mobile phones. In *DIMACS Workshop on Usable Privacy and Security Software*, 2004.