RESEARCH

1. QR BASICS

🧬 What’s Inside a QR Code?

At its core, a QR code is a **grid of black and white squares** (modules), and the pattern of those squares **encodes data** using binary (0s and 1s).

The **data inside** a QR code can include:

- URLs (most common)
- Text (up to ~4,000 characters)
- Phone numbers
- Email addresses
- Wi-Fi credentials
- Payment info
- Calendar events
- App deep links

🔢 How QR Codes Store Data: A Simple Breakdown

1. Binary Encoding

Data (like a URL) is first **converted into binary**—a string of 1s and 0s.

Example:
A simple URL like `https://example.com` becomes a binary string (just like digital computers use).

2. Structured Grid

The binary data is then encoded into a 2D square grid, where:

- **Black squares = 1**
- **White squares = 0**

The grid size depends on the **version** (there are 40 versions).
More data = bigger grid.

Version	Grid Size	Max Characters
1	21 x 21	~25 characters
10	57 x 57	~200 characters
40	177 x 177	~4,000+ characters

3. Error Correction

QR codes use **Reed-Solomon error correction**, which adds redundant data so the code still works even if part of it is:

- Dirty
- Torn
- Obscured by a logo

It can recover up to **30% of lost data** depending on the level used (L, M, Q, H).

---

4. Finder & Alignment Patterns

Certain parts of the QR code are **not data**, but are used to help scanners:

- **Finder patterns**: The three large squares in corners help the scanner know orientation.
- **Alignment patterns**: Help align the code if it's tilted.
- **Timing patterns**: Help detect the grid’s size and shape.

Here's a labeled diagram (imagine a QR code layout):

```
sql
CopyEdit
+-----------------------+
| ###       ###         | ← Finder patterns
| # #       # #         |
| ###       ###         |
|                      o| ← Alignment pattern
|                      |
|----------------------|
| Timing / Grid Guides |
+-----------------------+
```

🧠 In Short

A QR code is a **visually encoded version of digital data**—just a pattern of black/white modules that encode binary data + helper structures to ensure reliable scanning.

2. QR IN SIMPLE TERM

A **QR code is a visual representation of binary data** — that is, 1s and 0s — shown using **black and white squares** on a grid.

Let's break that down with a bit more clarity:

🔁 Text → Binary → Visual Squares

Example: Say you want to store the word "Hi"

1. **Text:** `"Hi"`
2. **Convert to Binary (ASCII):**
   - H = 72 = `01001000`
   - i = 105 = `01101001`
   - Full binary: `01001000 01101001`
3. **This binary data is then placed into a grid**, like this simplified example:

0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	1

4. **Displayed Visually:**
   - `1` = black square
   - `0` = white square

📷 When you scan that visual pattern with your phone, it reads the black/white modules, decodes them back into binary, then converts that binary back into text, a URL, etc.

🧱 And It's Not Just Data…

Besides the actual message, the QR code **also visually includes:**

- **Error correction data** (to fix damage)
- **Metadata** (like encoding mode, version)
- **Helper structures** (like finder and alignment patterns)

✅ Bottom Line

Yes — the QR code is **binary data made visible**, in the form of a carefully designed black-and-white square grid that can be read optically by scanners.

3. PROBLEMS IN A URL

A **URL can be a carrier for all sorts of malicious or deceptive behavior**, especially when hidden inside something like a QR code where you can't *see* the link upfront.

Here's a clear list of the **common risks and threats** that can hide inside a URL:

🧨 Risks Hidden in a URL

1. Phishing Links

- Fake websites that look like real ones (e.g., bank login pages) to **steal passwords**, credit card numbers, or personal info.

2. Malware Downloads

- Direct links to **infected files** (e.g., EXE, APK, PDF) that install malware, spyware, or ransomware on your device.

3. Drive-By Downloads

- Just **visiting the page** triggers an automatic file download or a browser exploit — no click required.

4. Credential Harvesters

- Login forms on fake websites that **record your username/password** and send them to attackers.

5. Command Execution (for vulnerable systems)

- URLs crafted to **exploit server-side flaws** or browser vulnerabilities (e.g., JavaScript injection or XSS attacks).

6. Browser Exploits

- URLs can contain scripts or embedded content that exploit **bugs in your browser or plugins** (e.g., Flash, old Java).

7. Fake App Downloads

- Especially with mobile QR codes: links to **unofficial app stores** or APKs that mimic real apps but are infected.

8. Credential Stuffing & Tracking

- URLs with **tracking tokens** or **session-stealing scripts** can hijack logins or gather sensitive user behavior.

9. Scam Pages or Fake Ads

- “Congratulations! You won an iPhone!” pages meant to **harvest info** or trick you into subscriptions/payments.

10. Shortened or Obfuscated Links

- URLs from services like `bit.ly` or `tinyurl.com` hide the true destination, often used to mask dangerous sites.

11. IP Logging & Tracking

- Even a harmless-looking link may be designed to **collect your IP address**, location, or browser fingerprint.

🔐 Tips to Stay Safe:

- ✅ **Preview QR code links** (some apps show the full URL before opening it).
- ✅ Use **secure browsers** and keep them updated.
- ✅ Avoid scanning QR codes in sketchy places (e.g., random flyers, public bathrooms).
- ✅ Use **QR code scanners** with built-in safety features (like previewing or scanning the destination first).
- ✅ Watch for misspelled domains (e.g., `paypa1.com` instead of `paypal.com`).

4. GSB API SOLVABLES

Google Safe Browsing API is a powerful tool, but it's not **all-seeing**. Here's a breakdown of what **Google Safe Browsing *can*** detect well, and what it **can't reliably catch**, especially in real time or in edge cases.

✅ What Google Safe Browsing *CAN* Detect

It works best at identifying:

1. **Known Phishing Sites**
   - Fake login pages, banking scams, or credential harvesters listed in its databases.
2. **Malware Hosting URLs**
   - Sites distributing known malicious software (e.g., trojans, ransomware, spyware).
3. **Social Engineering/Scam Sites**
   - Fake giveaways, deceptive prompts ("Your iPhone is infected!"), etc.
4. **Unsafe Downloads**
   - Links to files flagged as dangerous by Google's analysis (executables, ZIPs, etc.).

⚠️ What Google Safe Browsing *CANNOT Reliably Detect*

(or detect *only after damage is done*)

🧩 1. Zero-Day Malware or Newly Created Threats

- URLs that have **not yet been reported or discovered**.
- There's a lag between when the malicious link appears and when Google flags it.

🔒 2. Password Phishing on Obscure or Customized Domains

- Sophisticated attackers can **mimic real websites** using typosquatting (e.g., `g00gle.com`) or custom short domains that aren't in Google's blacklist yet.

📦 3. Drive-By Attacks Using Browser Exploits

- Unless the site is *already* known to do this, Google won't always catch it preemptively.

🐍 4. Script Injection or Obfuscated JavaScript

- A URL that links to a legitimate-looking site that loads malicious scripts from other servers can bypass detection.

🔍 5. Links Hidden Behind URL Shorteners

- Google may not resolve all short links (e.g., `bit.ly`, `t.co`, etc.) unless you expand and check them first.

🧬 6. Targeted Tracking & IP Logging

- A URL might be used to **track your IP, browser info, or location** — Google doesn't treat that as "malicious" unless abuse is proven.

🛠️ 7. Malicious Use of Legitimate Services

- Attackers often use **Google Docs, Dropbox, Firebase, etc.** to deliver malicious content. Safe Browsing might not flag these because they come from trusted domains.

🦠 8. Non-URL Payloads (e.g., QR code contains Wi-Fi config)

- If a QR code configures a Wi-Fi network, sends an SMS, or creates a calendar invite — those aren't URLs and bypass Safe Browsing checks entirely.

🚨 Key Insight:

Google Safe Browsing is reactive, not proactive. It's like a blacklist: it works only once a site is already reported or analyzed.

So even with it, you should:

- Be cautious with **unknown QR codes or links.**
- Use **link expanders** and **preview tools**.
- Cross-check domains manually for suspicious typos or redirects.

---

5. EXISTING SYSTEMS

There **are systems** (both websites and tools) that let you check if a **URL is safe**, and there are even some that can scan **QR codes without opening the link**.

🔍 🛡️ Part 1: Tools to Check if a URL is Safe

These services analyze a URL and tell you if it's flagged as **malicious, phishing, scammy, or suspicious**:

✅ Popular URL Safety Checkers:

1. **Google Safe Browsing Site Status**
   - Paste the URL to see if it's known as harmful.
2. **VirusTotal**
   - Scans URLs (and files) using **70+ antivirus engines** and security tools.
   - Also gives details like domain age, IP, location, behavior, etc.
3. **URLVoid**
   - Checks reputation of domains using multiple security databases.
4. **PhishTank**
   - Focuses on phishing URLs — lets you check/report suspicious links.

📷 🧾 Part 2: Tools to Scan a QR Code Without Opening It

Yes, this is possible — these tools **extract and display the content of a QR code** (URL, text, config) **without activating or opening it**, so you stay safe.

✅ Ways to Do This:

🖥️ Web Tools

1. **ZXing Decoder Online**
   - Upload an image of the QR code.
   - It will **decode and show the content** (like a URL or Wi-Fi info), without executing it.
2. **QRStuff Decoder**
   - Another visual decoder — shows raw content inside a QR.
3. **[VirusTotal + QR code]**
   - First decode the QR with ZXing or similar.
   - Then copy the extracted URL into VirusTotal to scan it for threats.

📱 Mobile Apps (Safer Scanners)

- **Kaspersky QR Scanner**
- **Norton Snap QR Reader**
- **Trend Micro QR Scanner**

These apps:

- Show you the decoded content first.
- Warn you if it points to a known malicious site.
- Don’t open the link automatically.

🧠 Best Practice Flow for Safety:

1. **Scan QR code with a safe reader or decoder** (no auto-opening).
2. **Copy the URL** it contains.
3. **Check the URL** using VirusTotal, Google Safe Browsing, or URLVoid.
4. **Then decide** whether to visit or ignore.

END