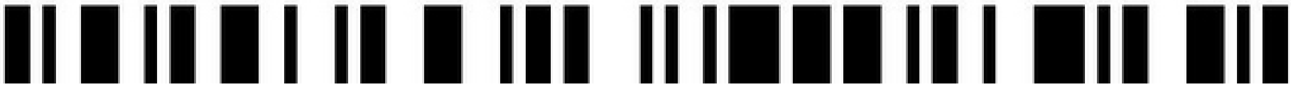
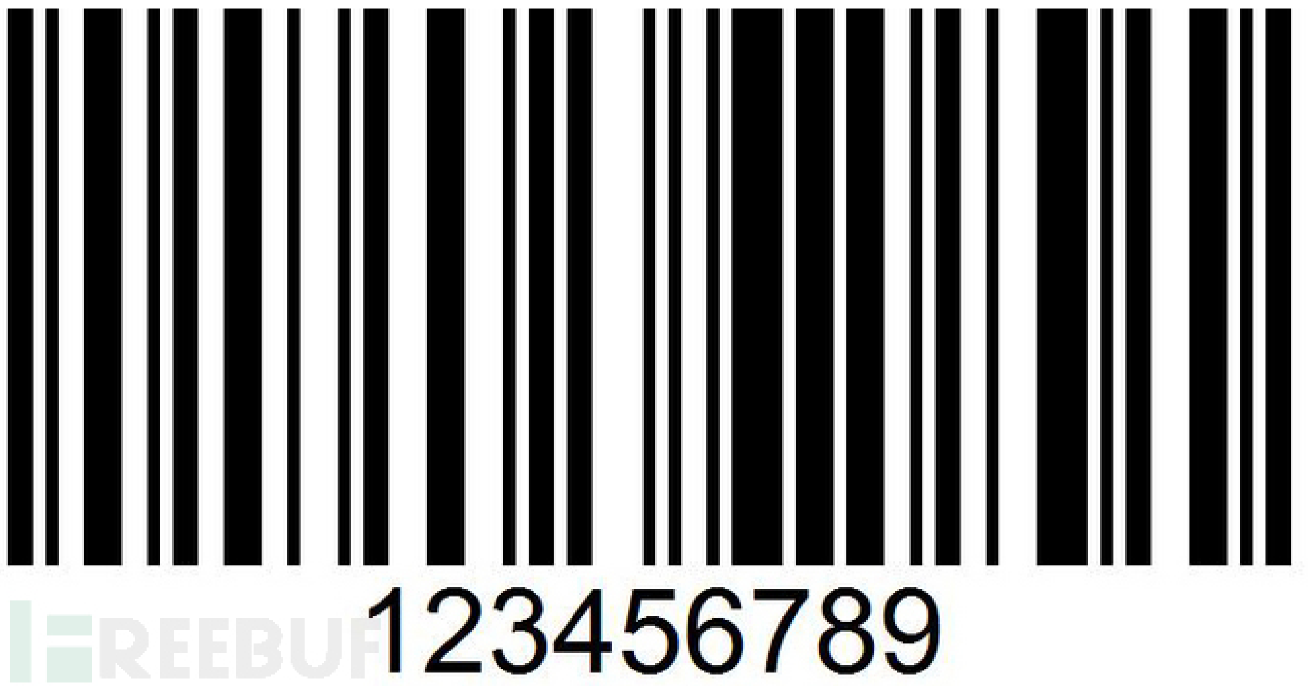
原创作者：阿金

在GEEK PWN大会上，TK为大家演示了腾讯玄武实验室最近研究的新成果—badbarcode，也就是对条码中一些安全隐患的研究。其实关于条码及二维码的安全研究由来已久，本文主要为大家介绍一下badbarcode最基本的原理以及简单的badbarcode攻击。

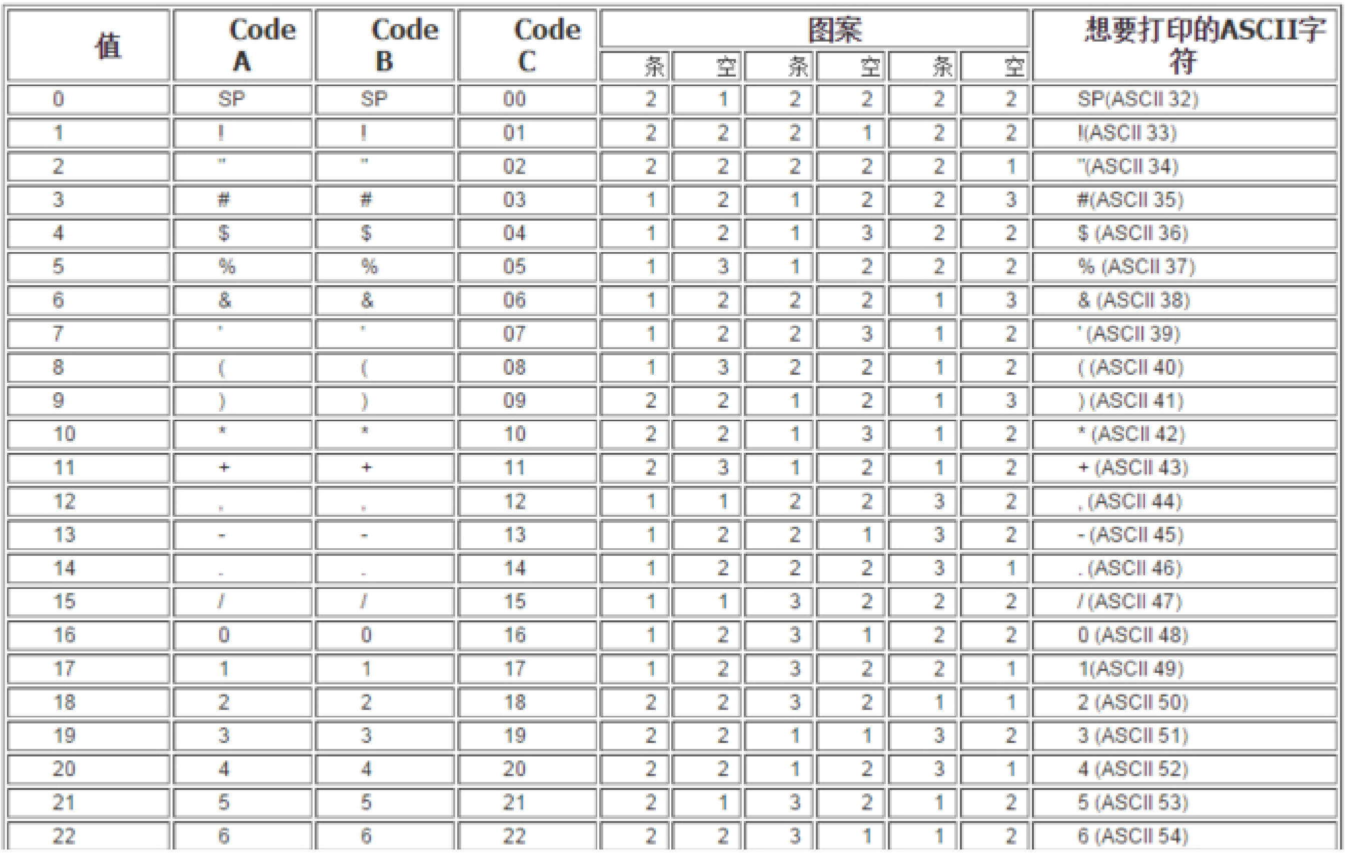
在badbarcode中，主要针对的是code128协议，由于几乎所有的读码器都可以识别code128协议的条码，因此，此方法还是十分通用的。Code128码其实还分成三类，即128A、128B、128C。它们的区别就是对应的字符表不一样。那么code128码到底怎么读呢？请看下面的图：





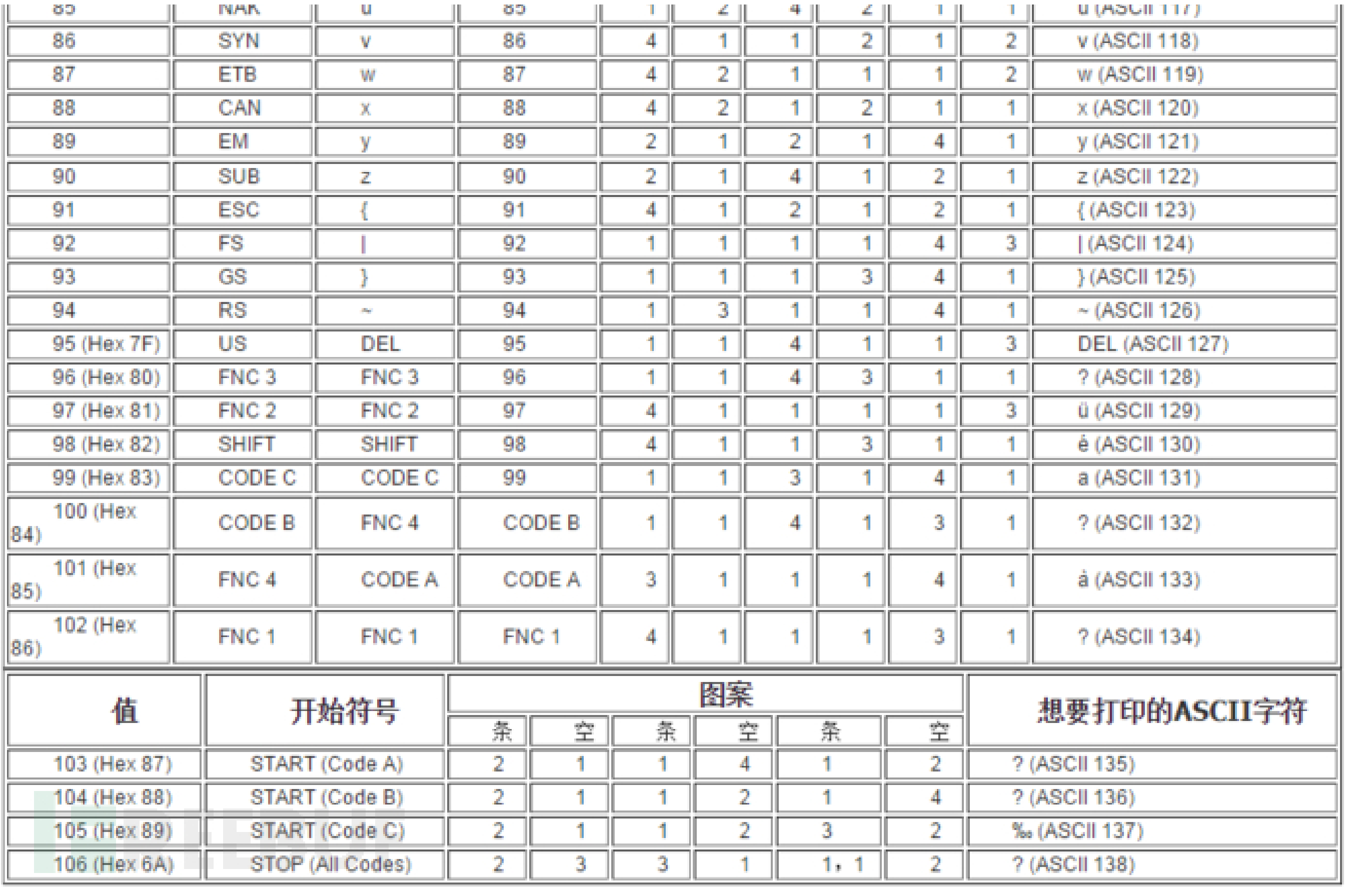
条码中从左往右是黑白相间的条形图，其中黑颜色的叫做“条”，用B表示，白颜色的叫做“空”，用S表示。条和空都有4种不同的宽度，我们将其从细到粗赋予1、2、3、4四个值。然后我们开始按照不同粗细的值来阅读上面的条码，结果为：211232 112232 131123 331121 241112 214121 124211 2331112。

Code128码有一个头一个尾。尾用2331112来表示，这代表Code128已经结束；尾前面的6位是校验位，用于检查该条形码是否被正确编码；头有3种，分别是211412表示128A、211214表示128B、211232表示128C；其余的部分是6位为一个块。去掉头、尾以及校验码后就是这样：112232 131123 331121 241112 214121。我们可以根据字符表读出该条码的具体内容，字符表如下所示：



值	Code A	Code B	Code C	图案						想要打印的ASCII字符
				条	空	条	空	条	空	
0	SP	SP	00	2	1	2	2	2	2	SP(ASCII 32)
1	!	!	01	2	2	2	1	2	2	!(ASCII 33)
2	"	"	02	2	2	2	2	2	1	"(ASCII 34)
3	#	#	03	1	2	1	2	2	3	#(ASCII 35)
4	$	$	04	1	2	1	3	2	2	$ (ASCII 36)
5	%	%	05	1	3	1	2	2	2	% (ASCII 37)
6	&	&	06	1	2	2	2	1	3	& (ASCII 38)
7	'	'	07	1	2	2	3	1	2	' (ASCII 39)
8	(	(	08	1	3	2	2	1	2	((ASCII 40)
9	)	)	09	2	2	1	2	1	3	) (ASCII 41)
10	*	*	10	2	2	1	3	1	2	* (ASCII 42)
11	+	+	11	2	3	1	2	1	2	+ (ASCII 43)
12	,	,	12	1	1	2	2	3	2	, (ASCII 44)
13	-	-	13	1	2	2	1	3	2	- (ASCII 45)
14	.	.	14	1	2	2	2	3	1	. (ASCII 46)
15	/	/	15	1	1	3	2	2	2	/ (ASCII 47)
16	0	0	16	1	2	3	1	2	2	0 (ASCII 48)
17	1	1	17	1	2	3	2	2	1	1(ASCII 49)
18	2	2	18	2	2	3	2	1	1	2 (ASCII 50)
19	3	3	19	2	2	1	1	3	2	3 (ASCII 51)
20	4	4	20	2	2	1	2	3	1	4 (ASCII 52)
21	5	5	21	2	1	3	2	1	2	5 (ASCII 53)
22	6	6	22	2	2	3	1	1	2	6 (ASCII 54)

23	7	7	23	3	1	2	1	3	1	7 (ASCII 55)
24	8	8	24	3	1	1	2	2	2	8 (ASCII 56)
25	9	9	25	3	2	1	1	2	2	9 (ASCII 57)
26	:	:	26	3	2	1	2	2	1	: (ASCII 57)
27	;	;	27	3	1	2	2	1	2	; (ASCII 59)
28	<	<	28	3	2	2	1	1	2	< (ASCII 60)
29	=	=	29	3	2	2	2	1	1	= (ASCII 61)
30	>	>	30	2	1	2	1	2	3	> (ASCII 62)
31	?	?	31	2	1	2	3	2	1	? (ASCII 63)
32	@	@	32	2	3	2	1	2	1	@ (ASCII 64)
33	A	A	33	1	1	1	3	2	3	A (ASCII 65)
34	B	B	34	1	3	1	1	2	3	B (ASCII 66)
35	C	C	35	1	3	1	3	2	1	C (ASCII 67)
36	D	D	36	1	1	2	3	1	3	D (ASCII 68)
37	E	E	37	1	3	2	1	1	3	E (ASCII 69)
38	F	F	38	1	3	2	3	1	1	F (ASCII 70)
39	G	G	39	2	1	1	3	1	3	G (ASCII 71)
40	H	H	40	2	3	1	1	1	3	H (ASCII 72)
41	I	I	41	2	3	1	3	1	1	I (ASCII 73)
42	J	J	42	1	1	2	1	3	3	J (ASCII 74)
43	K	K	43	1	1	2	3	3	1	K (ASCII 75)
44	L	L	44	1	3	2	1	3	1	L (ASCII 76)
45	M	M	45	1	1	3	1	2	3	M (ASCII 77)
46	N	N	46	1	1	3	3	2	1	N (ASCII 78)
47	O	O	47	1	3	3	1	2	1	O (ASCII 79)
48	P	P	48	3	1	3	1	2	1	P (ASCII 80)
49	Q	Q	49	2	1	1	3	3	1	Q (ASCII 81)
50	R	R	50	2	3	1	1	3	1	R (ASCII 82)
51	S	S	51	2	1	3	1	1	3	S (ASCII 83)
52	T	T	52	2	1	3	3	1	1	T (ASCII 84)
53	U	U	53	2	1	3	1	3	1	U (ASCII 85)
54	V	V	54	3	1	1	1	2	3	V (ASCII 86)
55	W	W	55	3	1	1	3	2	1	W (ASCII 87)
56	X	X	56	3	3	1	1	2	1	X (ASCII 88)
57	Y	Y	57	3	1	2	1	1	3	Y (ASCII 89)
58	Z	Z	58	3	1	2	3	1	1	Z (ASCII 90)
59	[	[	59	3	3	2	1	1	1	[(ASCII 91)
60	\	\	60	3	1	4	1	1	1	\ (ASCII 92)
61	]	]	61	2	2	1	4	1	1	] (ASCII 93)
62	^	^	62	4	3	1	1	1	1	^ (ASCII 94)
63	_	_	63	1	1	1	2	2	4	_ (ASCII 95)
64	NUL	`	64	1	1	1	4	2	2	` (ASCII 96)
65	SOH	a	65	1	2	1	1	2	4	a (ASCII 97)
66	STX	b	66	1	2	1	4	2	1	b (ASCII 98)
67	ETX	c	67	1	4	1	1	2	2	c (ASCII 99)
68	EOT	d	68	1	4	1	2	2	1	d (ASCII 100)
69	ENQ	e	69	1	1	2	2	1	4	e (ASCII 101)
70	ACK	f	70	1	1	2	4	1	2	f (ASCII 102)
71	BEL	g	71	1	2	2	1	1	4	g (ASCII 103)
72	BS	h	72	1	2	2	4	1	1	h (ASCII 104)
73	HT	i	73	1	4	2	1	1	2	i (ASCII 105)
74	LF	j	74	1	4	2	2	1	1	j (ASCII 106)
75	VT	k	75	2	4	1	2	1	1	k (ASCII 107)
76	FF	l	76	2	2	1	1	1	4	l (ASCII 108)
77	CR	m	77	4	1	3	1	1	1	m (ASCII 109)
78	SO	n	78	2	4	1	1	1	2	n (ASCII 110)
79	SI	o	79	1	3	4	1	1	1	o (ASCII 111)
80	DLE	p	80	1	1	1	2	4	2	p (ASCII 112)
81	DC1	q	81	1	2	1	1	4	2	q (ASCII 113)
82	DC2	r	82	1	2	1	2	4	1	r (ASCII 114)
83	DC3	s	83	1	1	4	2	1	2	s (ASCII 115)
84	DC4	t	84	1	2	4	1	1	2	t (ASCII 116)
85	NAK	u	85	1	2	4	2	1	1	u (ASCII 117)

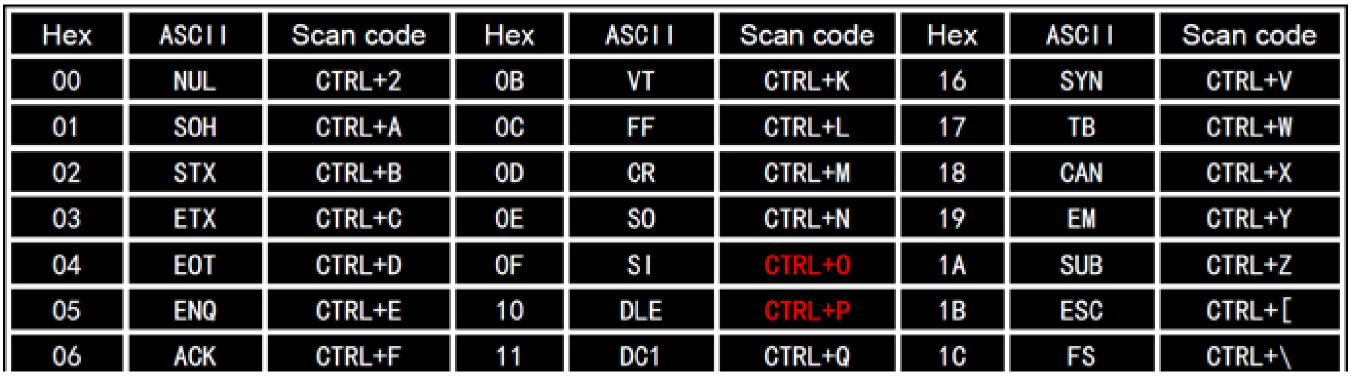


85	NAK	u	85	1	2	4	2	1	1	u (ASCII 117)
86	SYN	v	86	4	1	1	2	1	2	v (ASCII 118)
87	ETB	w	87	4	2	1	1	1	2	w (ASCII 119)
88	CAN	x	88	4	2	1	2	1	1	x (ASCII 120)
89	EM	y	89	2	1	2	1	4	1	y (ASCII 121)
90	SUB	z	90	2	1	4	1	2	1	z (ASCII 122)
91	ESC	{	91	4	1	2	1	2	1	{ (ASCII 123)
92	FS	|	92	1	1	1	1	4	3	| (ASCII 124)
93	GS	}	93	1	1	1	3	4	1	} (ASCII 125)
94	RS	~	94	1	3	1	1	4	1	~ (ASCII 126)
95 (Hex 7F)	US	DEL	95	1	1	4	1	1	3	DEL (ASCII 127)
96 (Hex 80)	FNC 3	FNC 3	96	1	1	4	3	1	1	? (ASCII 128)
97 (Hex 81)	FNC 2	FNC 2	97	4	1	1	1	1	3	ü (ASCII 129)
98 (Hex 82)	SHIFT	SHIFT	98	4	1	1	3	1	1	é (ASCII 130)
99 (Hex 83)	CODE C	CODE C	99	1	1	3	1	4	1	a (ASCII 131)
100 (Hex 84)	CODE B	FNC 4	CODE B	1	1	4	1	3	1	? (ASCII 132)
101 (Hex 85)	FNC 4	CODE A	CODE A	3	1	1	1	4	1	à (ASCII 133)
102 (Hex 86)	FNC 1	FNC 1	FNC 1	4	1	1	1	3	1	? (ASCII 134)

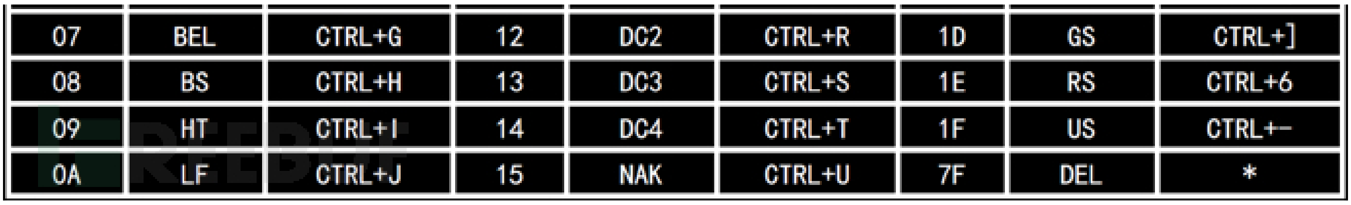
值	开始符号	图案						想要打印的ASCII字符
		条	空	条	空	条	空	
103 (Hex 87)	START (Code A)	2	1	1	4	1	2	? (ASCII 135)
104 (Hex 88)	START (Code B)	2	1	1	2	1	4	? (ASCII 136)
105 (Hex 89)	START (Code C)	2	1	1	2	3	2	‰ (ASCII 137)
106 (Hex 6A)	STOP (All Codes)	2	3	3	1	1，1	2	? (ASCII 138)

根据头我们可以知道这个是Code128C型。C型码是纯数字的，每个块对应2位数字，查表可得：12 34 5 6 78 90

要研究badbarcode，我们就要了解一下条码扫描仪是如何识别条码并对条码中的数据进行处理的。识别条码的原理在上面已经介绍过了，那么识别出其中的数据后扫码仪是如何工作的呢？其实大部分扫码仪是基于模仿键盘的机制进行工作的，也就是说在条码中读出数据后，机器会认为该数据是通过键盘输入的，从而在商品编号等信息框内输入商品信息。那么这样的话，我们可不可以通过控制条码信息来在电脑上执行命令呢？答案当然是可以的！在code128协议中，是支持ASCII控制字符的。也就是说在code128协议中对于输入的ASCII字符都会有一个控制字符与之对应，具体的对应情况请看下表：

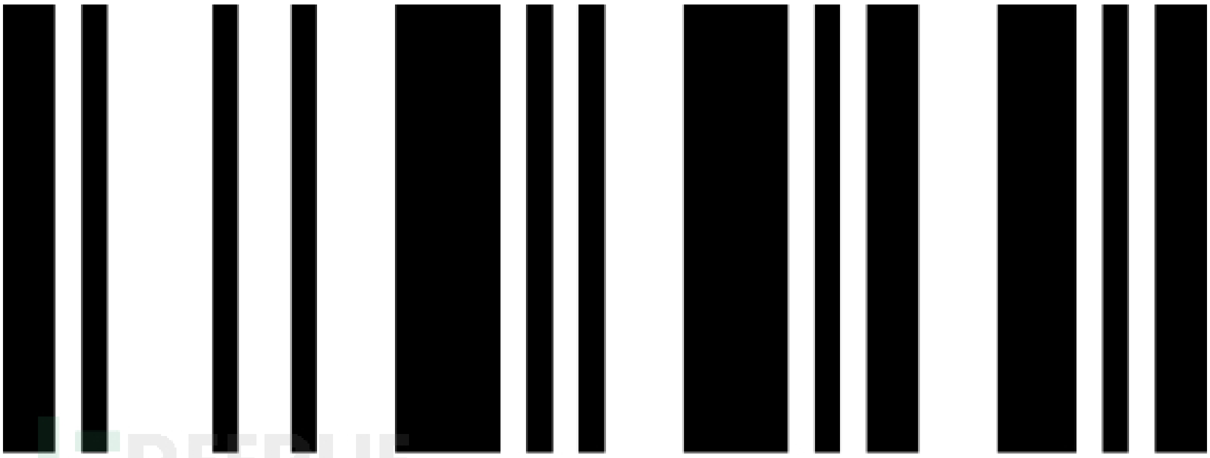


Hex	ASCII	Scan code	Hex	ASCII	Scan code	Hex	ASCII	Scan code
00	NUL	CTRL+2	0B	VT	CTRL+K	16	SYN	CTRL+V
01	SOH	CTRL+A	0C	FF	CTRL+L	17	TB	CTRL+W
02	STX	CTRL+B	0D	CR	CTRL+M	18	CAN	CTRL+X
03	ETX	CTRL+C	0E	SO	CTRL+N	19	EM	CTRL+Y
04	EOT	CTRL+D	0F	SI	CTRL+O	1A	SUB	CTRL+Z
05	ENQ	CTRL+E	10	DLE	CTRL+P	1B	ESC	CTRL+[
06	ACK	CTRL+F	11	DC1	CTRL+Q	1C	FS	CTRL+\

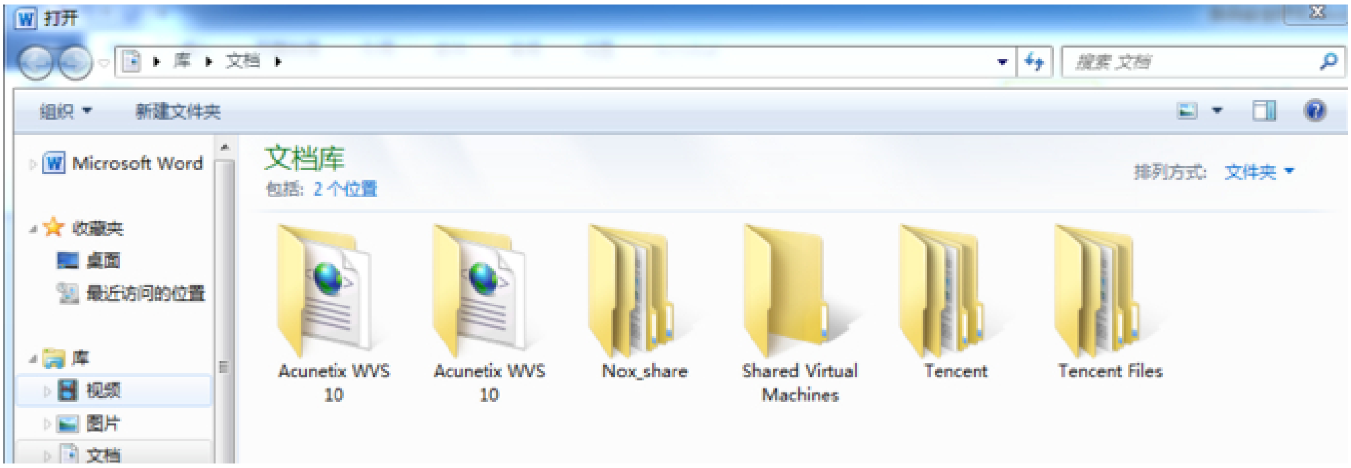


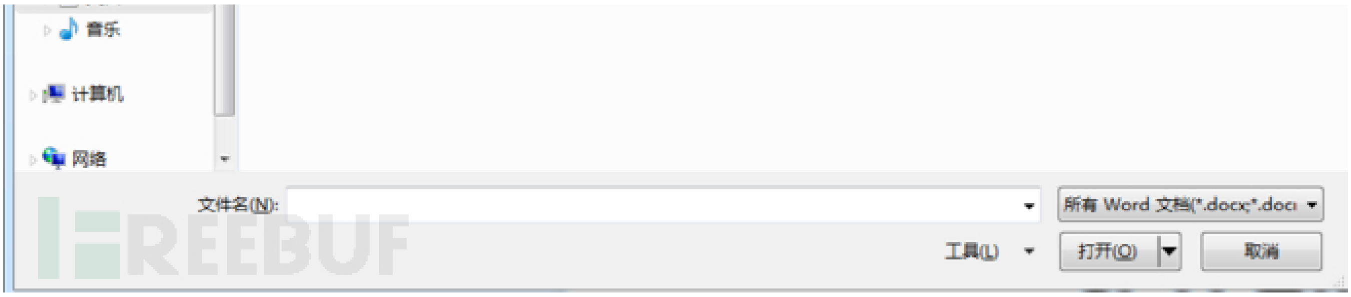
07	BEL	CTRL+G	12	DC2	CTRL+R	1D	GS	CTRL+]
08	BS	CTRL+H	13	DC3	CTRL+S	1E	RS	CTRL+6
09	HT	CTRL+I	14	DC4	CTRL+T	1F	US	CTRL+–
0A	LF	CTRL+J	15	NAK	CTRL+U	7F	DEL	*

通过上表我们可以看到，针对不同的ASCII字符，都有一个CTRL+X(X代表任意键)与之对应，这样呢，我们就可以通过执行CTRL+O打开对话框，然后大家就可以继续使用一些方式来进行进一步的操作了。根据上面对code128协议的介绍，我们可以通过生成一个含有2114121341111341112331112信息的条码来执行CTRL+O命令。该信息表明此条码是128A类型的条码，其中包含ASCII字符SI，即对应着CTRL+O命令。生成该条码后，条码如下所示：



扫码仪扫描过上面条码后就会打开如下窗口：





以上只是badbarcode中最基础的内容，在此仅做一个简单的解析，更加精彩的攻击方式可以通过ADF技术自定义键盘输入来实现构造一段包含任意命令的条码。具体关于ADF的技术原理在此就不做介绍了，有兴趣的可以自己进行进一步的研究。

***原创作者：阿金，本文属FreeBuf原创奖励计划文章，未经许可禁止转载**