Snooper

APDU script program language

Programmer's Reference Manual

About this document

为实现 apdu 脚本简单化以及规范化，taoism 6 版对常规版的脚本工具进行了较大修改，本手册中涉及的软件为 snoopers taoism 6 版，适用于 0.0.6.7 以上版本，本手册为第 4 部分，脚本语法、使用以及函数列表，请见第 1、2、3 部分。

目　录

简介

基本介绍

Snooper是为智能卡应用开发者提供测试数据的工具。Snooper的开发过程是随着智能卡应用开发需求的不断变化逐步完善的过程。Snooper脚本具有独特的风格，注重简单、可靠的脚本编写体验，支持动态编辑执行、自动计算、流程跳转、动态提示、自动完成、自定义过程等多种功能，积累了丰富的测试脚本。使用Snooper工具，可以高效进行智能卡开发与测试。

更新变化

0.0.6.7----增加while-loop循环支持，与do-loop循环类似，更简洁。
0.0.6.7----增加new_rsa_get_n, new_ecc_get_pri等函数，用于简化脚本。
0.0.6.7----增加了 pqc 相关的函数
0.0.6.7----增加了 resume 关键字，可以转到通讯错误的行号前后进行处理
0.0.6.7----在脚本编辑器中提示函数参数时，对可选参数使用方括号标识。
0.0.6.7----增加了 aes cmac 计算函数，支持 init, update, final 模式，其他算法暂未支持。
0.0.6.7----增加了简单的变量指针操作，使用&标号取得变量指针，并通过 get_value 和 set_value 来读写
0.0.6.7----增加了 neg 函数，变量求负数值。
0.0.6.7----增加了$$取指针变量对应的值。
0.0.6.7----增加了 v2a, v2u 两个宏定义，用于输出 log 时方便。
0.0.6.8----增加了 x64 版本的 exe。
0.0.6.8----增加了 setup_makebat 函数，用于删除编译时生成的临时文件。
0.0.6.8----增加了 hidpi 的简单支持功能。
0.0.6.8----增加了 Build_5f2e_face_by_mem 和 Build_5f2e_face_by_file。

使用注意

1） 本程序是一个测试工具，有一定的编辑功能，日常情况下可以满足编辑要求，也可使用其他专业工具来编辑脚本，使用本程序执行脚本。

2） 本程序是一个类似多文档的工具，可以同时打开多个脚本，但是只存在一个工作环境，工作环境中的多个脚本，逻辑上是统一的，允许互相调用。

3） 第一个被打开的称为主脚本，主脚本中可以使用 load_script 来打开辅助脚本。

4） load_script 语句只有在主脚本中对其他辅助脚本才有效。

5） load_script 语句只有在打开主脚本时检测，动态编辑的无效。

6） 主脚本和load_script 语句中所有涉及的脚本（无论是否存在）都会被工具监控是否被改动。

7） dummy函数，dummy函数是一个api接口，可调用snooper中的另一组函数，新的函数支持多层嵌套，函数名称与当前脚本函数有极大部分名称相同，所有使用时要仔细区分。

8） 一般情况下，打开新脚本时会检查现有脚本是否被改动，并提示保存，用户应按实际需求决定是否保存脚本。

9） 脚本编辑窗口具有定时保存临时文件功能。

Resume 关键字

```
on error goto err_handle

tt  = 00

0088000088


end

err_handle:

    tt  = add( $tt, 01 )
    if $tt == 01
        resume -1
    else if $tt == 02
        resume
    else
        resume 1
    endif
```

变量指针操作

```
clear

t1  = 99999999
t2  = ""
```

```
t3   = 8888

call test_pointer( $t1, &t2, $t3 )

? $t1
? $t2
? $t3
clear

a    = 99
c    = &a

// 指针变量未定义，无法取值
b    = $$d

// 指针变量内容错误，无法取值
e    = 88888888999899999999999999999
b    = $$e

// 指针变量内容格式正确，但是索引不正确
f    = add00088
b    = $$f

b    = $$c

end

test_pointer:
    a    = getpara
    b    = getpara
    c    = getpara

    a    = random( 2 )

    x    = set_value( $b, $a )

return
```

V2A、V2U

```
clear

k    = random( 1 )
```

```
t    = random( 1 )

if $k < 80
    s1  = "k < 0x80"
else
    s1  = "k >= 0x80"
endif

if $t < 80
    s2  = utf16_little_string( "t < 0x80" )
else
    s2  = utf16_little_string( "t >= 0x80" )
endif

? "k信息 " v2a( $s1 )
? "t信息 " v2u( $s2 )
```

0.0.6.7

new_rsa_get_n

new_rsa_get_d

new_rsa_get_p

new_rsa_get_q

new_rsa_get_dp

new_rsa_get_dq

new_rsa_get_qinv

用法	

```
x          = new_rsa_gencrt( $len, $e )
n1         = hmid( $x, 00, $len_byte )
d1         = hmid( $x, $len_byte, $len_byte )
p1         = hmid( $x, hex( 0x$plen_byte * 4 ), $plen_byte )
q1         = hmid( $x, hex( 0x$plen_byte * 5 ), $plen_byte )
dp1        = hmid( $x, hex( 0x$plen_byte * 6 ), $plen_byte )
dq1        = hmid( $x, hex( 0x$plen_byte * 7 ), $plen_byte )
qinv1      = hmid( $x, hex( 0x$plen_byte * 8 ), $plen_byte )
```

```
n2          = new_rsa_get_n()
d2          = new_rsa_get_d()
p2          = new_rsa_get_p()
q2          = new_rsa_get_q()
dp2         = new_rsa_get_dp()
dq2         = new_rsa_get_dq()
qinv2       = new_rsa_get_qinv()
if $n1 != $n2
    ?
    pause
endif
if $d1 != $d2
    ?
    pause
endif
```

new_ecc_get_prikey

new_ecc_get_pubkey

用法

修改了new_ecc_get_pubkey，如果有参数，则计算公钥如果无参数，则读取最后一个生成的公钥

```
pri2    = new_ecc_get_prikey()
pub2    = new_ecc_get_pubkey()

//Name    =    secp 256 k1
len     = 02 56
P       = FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FE FF FF FC 2F
A       = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B       = 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07
GX      = 79 BE 66 7E F9 DC BB AC 55 A0 62 95 CE 87 0B 07 02 9B FC DB 2D CE 28 D9 59 F2 81 5B 16 F8 17 98
GY      = 48 3A DA 77 26 A3 C4 65 5D A4 FB FC 0E 11 08 A8 FD 17 B4 48 A6 85 54 19 9C 47 D0 8F FB 10 D4 B8
N       = FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FE BA AE DC E6 AF 48 A0 3B BF D2 5E 8C D0 36 41 41
H       = 01
x       = new_ecc_initialize( $p, $a, $b, $gx, $gy, $n, $h, 0100 )
```

```
x       = new_ecc_generate_keypair()
pri1    = hmid( $x, 0, 20 )
pub1    = hmid( $x, 20, 40 )

pri2    = new_ecc_get_prikey()
pub2    = new_ecc_get_pubkey()

if $pri1 != $pri2
    ?
    pause
endif

if $pub1 != $pub2
    ?
    pause
endif
```

fips203_keygen_512

fips203_keygen_768

fips203_keygen_1024

fips203_get_ek

fips203_get_dk

fips203_get_en_sharekey

fips203_get_de_sharekey

fips203_encaps_512

fips203_encaps_768

fips203_encaps_1024

fips203_decaps_512

fips203_decaps_768

fips203_decaps_1024

用法

```
d           = dup( 32, 11 )
z           = dup( 32, 22 )
```

```
m           = "this is my message"
// 填充到32字节
m           = pack00_len( $m, 20 )

x           = fips203_keygen_512( $d, $z )
if $x != 00
    ?
    pause
endif

ek          = fips203_get_ek()
dk          = fips203_get_dk()

c           = fips203_encaps_512( $m, $ek )
sharekey1   = fips203_get_en_sharekey()

m1          = fips203_decaps_512( $c, $dk )
sharekey2   = fips203_get_de_sharekey()

// 带检查的解密
m2          = fips203_decaps_512( $c, $dk, $ek )

if $m != $m1
    ?
    pause
endif

if $m != $m2
    ?
    pause
endif

if $sharekey1 != $sharekey2
    ?
    pause
endif
```

fips204_keygen_44

fips204_keygen_65

fips204_keygen_87

fips204_get_pk

fips204_get_sk

fips204_sign_44

fips204_sign_65

fips204_sign_87

fips204_sign_hash_44

fips204_sign_hash_65

fips204_sign_hash_87

fips204_verify_44

fips204_verify_65

fips204_verify_87

fips204_verify_hash_44

fips204_verify_hash_65

fips204_verify_hash_87

fips204_prehash_sha256

fips204_prehash_sha384

fips204_prehash_sha512

fips204_prehash_sha224

fips204_prehash_sha512_224

fips204_prehash_sha512_256

fips204_prehash_sha3_224

fips204_prehash_sha3_256

fips204_prehash_sha3_384

fips204_prehash_sha3_512

fips204_prehash_shake_128

fips204_prehash_shake_256

用法	clear
	
	seed = random(32)
	
	x = fips204_keygen_44($seed)
	if $x != 00
	?
	pause
	endif
	
	sk = fips204_get_sk()
	pk = fips204_get_pk()
	
	m = random(32)
	rnd = random(32)
	ctx = random(32)
	
	// 对明文操作
	sig = fips204_sign_44($m, $sk, $ctx, $rnd, "")
	
	res = fips204_verify_44($m, $sig, $pk, $ctx, "")
	if $res != 00
	?
	pause
	endif
	
	// 对hash操作
	h = fips204_prehash_sha256($m, $ctx)
	
	sig = fips204_sign_hash_44($h, $sk, $rnd, "")
	
	res = fips204_verify_hash_44($h, $sig, $pk, "")
	if $res != 00
	?
	pause
	endif

fips205_keygen_sha2__128s

fips205_keygen_shake_128s

fips205_keygen_sha2__128f

fips205_keygen_shake_128f

fips205_keygen_sha2__192s

fips205_keygen_shake_192s

fips205_keygen_sha2__192f

fips205_keygen_shake_192f

fips205_keygen_sha2__256s

fips205_keygen_shake_256s

fips205_keygen_sha2__256f

fips205_keygen_shake_256f

fips205_get_pk

fips205_get_sk

fips205_sign_sha2__128s

fips205_sign_shake_128s

fips205_sign_sha2__128f

fips205_sign_shake_128f

fips205_sign_sha2__192s

fips205_sign_shake_192s

fips205_sign_sha2__192f

fips205_sign_shake_192f

fips205_sign_sha2__256s

fips205_sign_shake_256s

fips205_sign_sha2__256f

fips205_sign_shake_256f

fips205_sign_hash_sha2__128s

fips205_sign_hash_shake_128s

fips205_sign_hash_sha2__128f

fips205_sign_hash_shake_128f

fips205_sign_hash_sha2__192s

fips205_sign_hash_shake_192s

fips205_sign_hash_sha2__192f

fips205_sign_hash_shake_192f

fips205_sign_hash_sha2__256s

fips205_sign_hash_shake_256s

fips205_sign_hash_sha2__256f

fips205_sign_hash_shake_256f

fips205_verify_sha2__128s

fips205_verify_shake_128s

fips205_verify_sha2__128f

fips205_verify_shake_128f

fips205_verify_sha2__192s

fips205_verify_shake_192s

fips205_verify_sha2__192f

fips205_verify_shake_192f

fips205_verify_sha2__256s

fips205_verify_shake_256s

fips205_verify_sha2__256f

fips205_verify_shake_256f

fips205_verify_hash_sha2__128s

fips205_verify_hash_shake_128s

fips205_verify_hash_sha2__128f

fips205_verify_hash_shake_128f

fips205_verify_hash_sha2__192s

fips205_verify_hash_shake_192s

fips205_verify_hash_sha2__192f

fips205_verify_hash_shake_192f

fips205_verify_hash_sha2__256s

fips205_verify_hash_shake_256s

fips205_verify_hash_sha2__256f

fips205_verify_hash_shake_256f

用法	

```
clear

sk_seed = random( 16 )
sk_prf  = random( 16 )
pk_seed = random( 16 )

rnd     = random( 16 )
m       = random( 32 )
ctx     = random( 32 )

// 生成密钥
x       = fips205_keygen_sha2__128f( $sk_seed, $sk_prf, $pk_seed )
if $x != 00
    ?
    pause
endif

sk      = fips205_get_sk()
pk      = fips205_get_pk()

// 对明文签名
sig     = fips205_sign_sha2__128f( $m, $sk, $ctx, $rnd )
```

```
res     = fips205_verify_sha2__128f( $m, $sig, $pk, $ctx )
if $res != 00
    ?
    pause
endif

// 对hash签名
h       = fips204_prehash_sha512( $m, $ctx )

sig     = fips205_sign_hash_sha2__128f( $h, $sk, $rnd )

res     = fips205_verify_hash_sha2__128f( $h, $sig, $pk )
if $res != 00
    ?
    pause
endif
```

aes128_cmac_init

aes128_cmac_update

aes128_cmac_final

aes192_cmac_init

aes192_cmac_update

aes192_cmac_final

aes256_cmac_init

aes256_cmac_update

aes256_cmac_final

des_cmac_init

des_cmac_update

des_cmac_final

3des_cmac_init

3des_cmac_update

3des_cmac_final

3des24_cmac_init

3des24_cmac_update

3des24_cmac_final

sm4_cmac_init

sm4_cmac_update

sm4_cmac_final

用法	clear ? "生成一个长度" begin: dec_indent

```
total_len    = random( 2 )
total_len    = and( $total_len, 01ff )
if $total_len <= 30
    goto begin
endif

? "生成一个数据"
data         = hrandom( $total_len )
icv          = hrandom( 10 )
key          = hrandom( 10 )

? "计算原始结果"
ori_res      = aes128_cmac( $icv, $data, $key )

? "使用循环，将data分成两段，做接力cmac计算"
off          = 00
prompt off
while $off < $total_len
    data1    = hleft( $data, $off )
    data2    = hmid( $data, $off )

    ? "分两包分别update"
    x        = aes128_cmac_init( $key, $icv )
    x        = aes128_cmac_update( $data1 )
    x        = aes128_cmac_update( $data2 )
    x        = aes128_cmac_final()
    if $x != $ori_res
        ?
        pause
    endif

    ? "分两包，最后一包在final中update"
    x    = aes128_cmac_init( $key, $icv )
    x    = aes128_cmac_update( $data1 )
    x    = aes128_cmac_final( $data2 )
    if $x != $ori_res
        ?
        pause
    endif

    ? "分两包，使用中间值"
    x            = aes128_cmac_init( $key, $icv )
    tmp_icv      = aes128_cmac_update( $data1 )
```

```
    整包长度     = big_div( $off, 10 )
    整包长度     = big_mul( $整包长度, 10 )
    缓存长度     = big_mod( $off, 10 )
    if $缓存长度 == 00
        // 这是因为最后一包只缓存，还没有参与计算
        缓存长度 = 10
    endif
    缓存数据     = hright( $data1, $缓存长度 )
    ? "重新初始化"
    x            = aes128_cmac_init( $key, $icv )
    x            = aes128_cmac_final( $data2, $tmp_icv, $整包长度, $缓存
数据 )
    if $x != $ori_res
        ?
        pause
    endif
    ? "重新初始化"
    x            = aes128_cmac_init( $key, $icv )
    x            = aes128_cmac_update( , $tmp_icv, $整包长度, $缓存数据 )
    x            = aes128_cmac_final( $data2 )
    if $x != $ori_res
        ?
        pause
    endif

    off = add( $off, 01 )
loop
prompt on
```

set_value

setvalue

get_value

getvalue

见变量指针操作

neg

用法

```
a = random( 8 )
b = neg( $a )
```

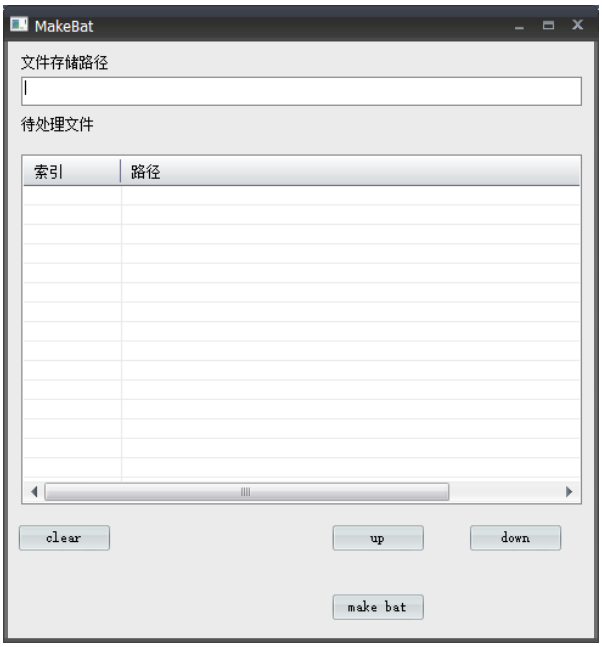
```
c = big_add( $a, $b )

a = random( 3 )
b = neg( $a, 01000000 )
c = big_add( $a, $b )
```

setup_makebat

用法

```
x = setup_makebat()
```



build_5f2e_face_by_mem

build_5f2e_face_by_file

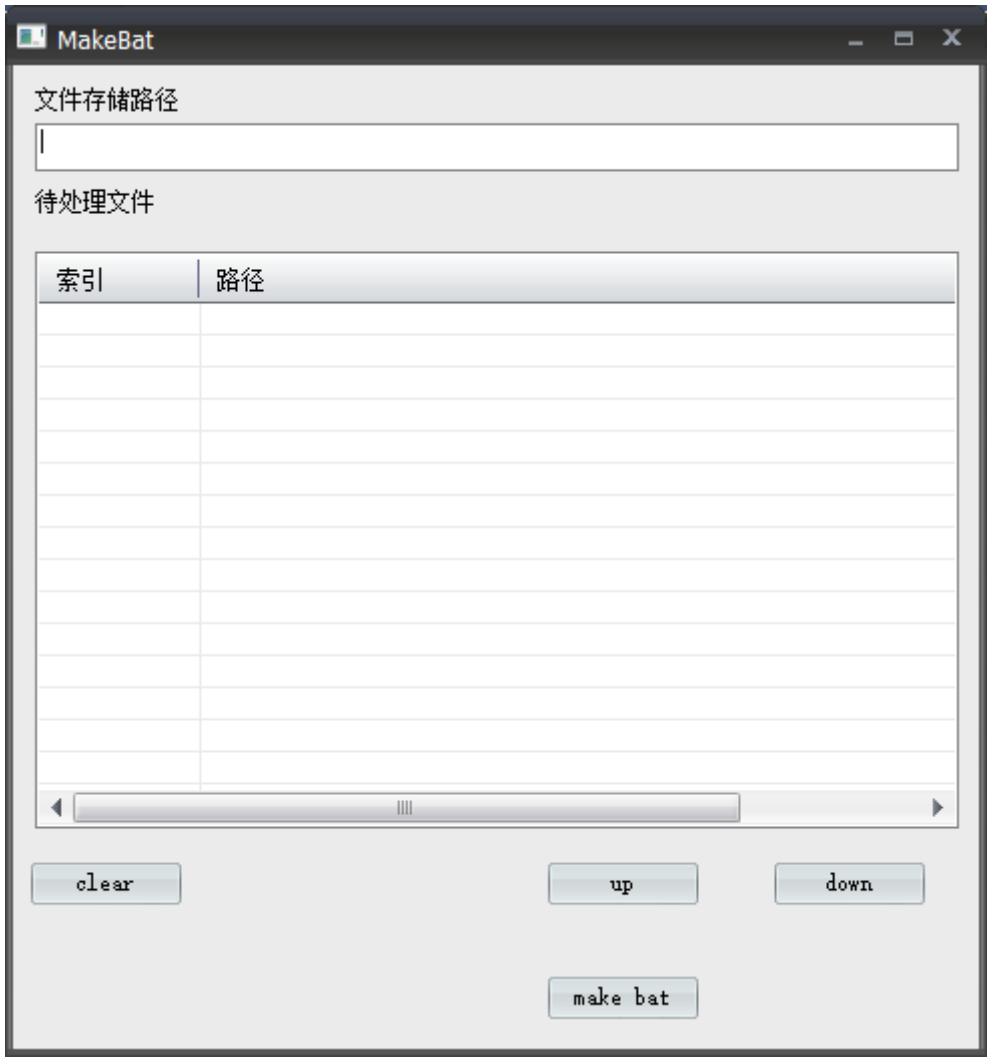
用法

```
d = file_readall( "logo.jpg" )

x = build_5f2e_face_by_mem( $d )

y = build_5f2e_face_by_file( "logo.jpg" )

if $x != $y
    ?
    pause
endif
```


MakeBat
文件存储路径
待处理文件
索引
路径
clear
up
down
make bat