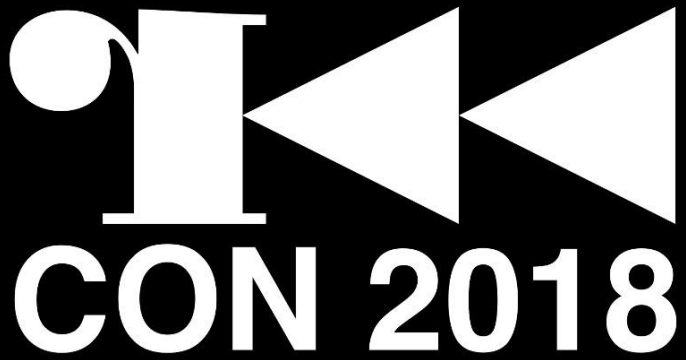
Recognition Techniques for Renaming Functions in Windows Malware

Matt Brooks, @cmatthewbrooks



Overview

Experiment - Hello, World!

- Old Windows development environment
  - Windows XP
  - Visual Studio 2005
  - Defaults (but compiled as C)

```c
#include <stdio.h>

int main(){
    printf("Hello, World!\n");
    return 0;
}
```

Experiment (IDA Freeware 5.0)

```c
#include <stdio.h>

int main(){
    printf("Hello, World!\n");
    return 0;
}
```

```
; int __cdecl main(int argc,const char **argv,const char *envp)
_main           proc near               ; CODE XREF: ___tmainCRTStartup+15A↓p
                push    offset aHelloWorld ; "Hello, World!\n"
                call    sub_401010
                add     esp, 4
                xor     eax, eax
                retn
_main           endp
```

```
sub_401010      proc near               ; CODE XREF: _main+5↑p

var_1C          = dword ptr -1Ch
ms_exc          = CPPEH_RECORD ptr -18h
arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch

                push    0Ch
                push    offset unk_40B390
                call    __SEH_prolog4
                xor     eax, eax
                xor     esi, esi
                cmp     [ebp+arg_0], esi
                setnz   al
                cmp     eax, esi
                jnz     short loc_401047
                call    __errno
                mov     dword ptr [eax], 16h
```

Experiment (IDA Freeware v7)

```c
#include <stdio.h>

int main(){
    printf("Hello, World!\n");
    return 0;
}
```

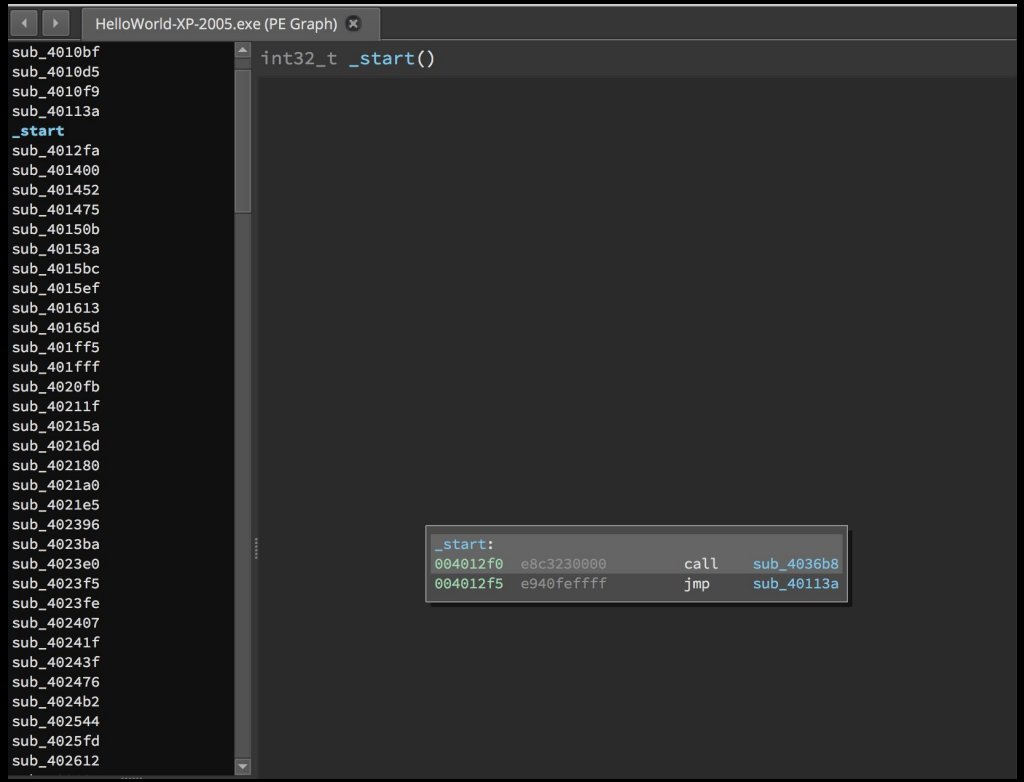
```
sub_401000      proc near               ; CODE XREF: start-5C↓p
                push    offset aHelloWorld ; "Hello, World!\n"
                call    sub_401010
                add     esp, 4
                xor     eax, eax
                retn
sub_401000      endp
```

```
                public start
start           proc near

; FUNCTION CHUNK AT .text:0040113A SIZE 00000030 BYTES
; FUNCTION CHUNK AT .text:0040116B SIZE 0000009C BYTES
; FUNCTION CHUNK AT .text:00401208 SIZE 00000010 BYTES
; FUNCTION CHUNK AT .text:00401219 SIZE 00000099 BYTES
; FUNCTION CHUNK AT .text:004012E0 SIZE 00000010 BYTES

                call    sub_4036B8
                jmp     loc_40113A
start           endp ; sp-analysis failed
```

Function name	Segment	Start
sub_401000	.text	0000000000401000
sub_401010	.text	0000000000401010
sub_4010BF	.text	00000000004010BF
sub_4010D5	.text	00000000004010D5
sub_4010F9	.text	00000000004010F9
start	.text	00000000004012F0
sub_4012FA	.text	00000000004012FA
sub_4013D1	.text	00000000004013D1
sub_401400	.text	0000000000401400
sub_401423	.text	0000000000401423
sub_401452	.text	0000000000401452
sub_401475	.text	0000000000401475
sub_40150B	.text	000000000040150B
sub_40153A	.text	000000000040153A
sub_4015BC	.text	00000000004015BC
sub_4015EF	.text	00000000004015EF
sub_401613	.text	0000000000401613
sub_40165D	.text	000000000040165D
sub_401FF5	.text	0000000000401FF5
sub_401FFF	.text	0000000000401FFF
sub_4020FB	.text	00000000004020FB
sub_40211F	.text	000000000040211F
sub_40215A	.text	000000000040215A
sub_40216D	.text	000000000040216D
sub_402180	.text	0000000000402180

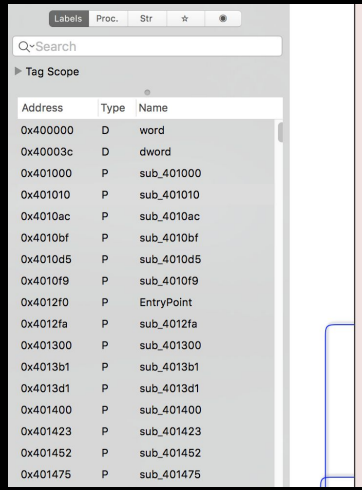
Experiment (Binary Ninja)

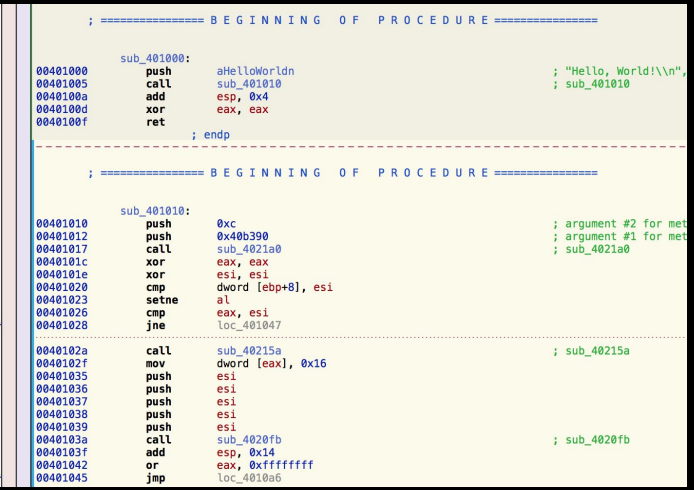
```c
#include <stdio.h>

int main(){
    printf("Hello, World!\n");
    return 0;
}
```



Experiment (Hopper)

```c
#include <stdio.h>

int main(){
    printf("Hello, World!\n");
    return 0;
}
```





Experiment (IDA Pro)

```c
#include <stdio.h>

int main(){
    printf("Hello, World!\n");
    return 0;
}
```

```
; int __cdecl main(int argc, const char **argv, const char **envp)
_main           proc near               ; CODE XREF: ___tmainCRTStartup+15A↓p

argc            = dword ptr  4
argv            = dword ptr  8
envp            = dword ptr  0Ch

                push    offset aHelloWorld ; "Hello, World!\n"
                call    _printf
                add     esp, 4
                xor     eax, eax
                retn
_main           endp
```

Problem

- Maximizing productivity when reversing Windows implants
- Many Visual Studio versions and C/C++ run-times
  - And MFC, ATL, etc…
  - Also - OpenSSL, sqlite, zlib, etc…
  - Oh yea, and Delphi.
- Dependence on FLIRT

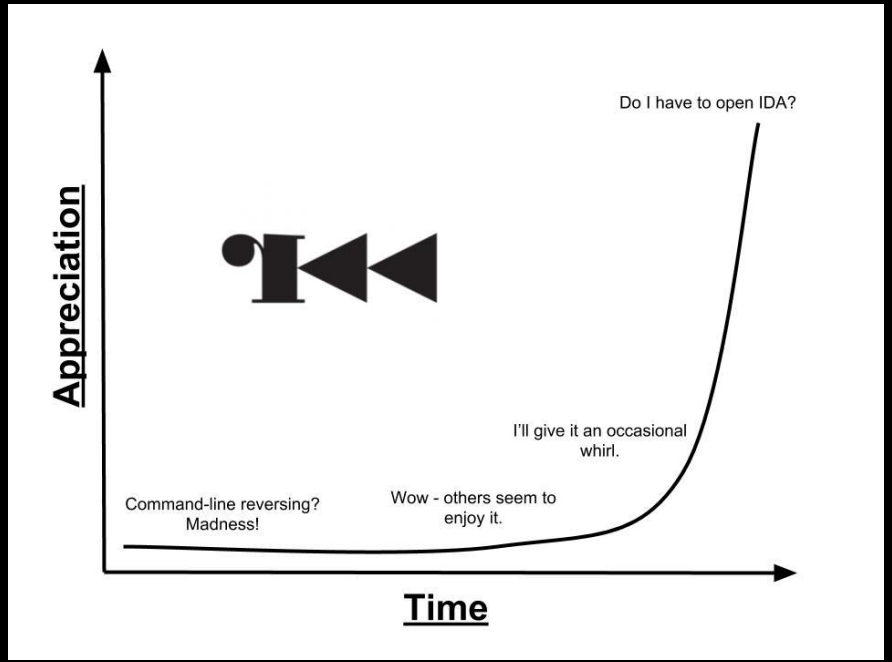
Implementations

FLIRT - IDA's Well-Known Solution

- Design
  - Parselib - First 32 "masked" bytes in a tree-based structure
  - Sigmake - Process patterns into different format
  - Signature files are compressed
- Application
  - Initial application based on special signature files for entry bytes

Implementation - sigs.py

- Implemented using r2pipe
- Multiple signature types
  - Zignature hash
  - String set hashes
  - Call chains (not quite yet)



Implementation - sigs.py

```
$ grep "class\|    def" sigs.py
class Matcher:
    The Matcher class is used to match signature hashes and rename fu
    def __init__(self, location, r2 = None):
    def match(self):
    def get_dict_key_from_value(self, dict, value):
class Maker:
    The Maker class is used to make signatures. It uses a Generator a
    def __init__(self, sigtype, location, outfile):
    def sigmake(self):
    def generate_func_name(self, name, location):
    def write_hash_file(self, outfile, sigtype):
    def validate_outfile(self, outfile):
class Generator(object):
    The Generator class generates function hashes of a given type. Ea
    type is implemented as a subclass where the subclass requirements
    a class attribute for the signature and a single method to return
    def __init__(self, r2):
        if str(r2.__class__) != 'r2pipe.open':
    def initialize_generators():
        for cls in Generator.__subclasses__():
    def generate(self, sigtype):
    def clear_hashes(self):
class ZigGenerator(Generator):
    def __init__(self, r2):
    def generate_hashes(self):
```

Explanation - Zignature Hashes

```
        0x10001c46      56              push esi
        0x10001c47      33f6            xor esi, esi
        0x10001c49      39742410        cmp dword [arg_10h], esi    ; [0x1d:4]=-1 ; 29
     ,=< 0x10001c4d      7e3b            jle 0x10001c8a
     |   0x10001c4f      8b54240c        mov edx, dword [arg_ch]     ; [0xc:4]=-1 ; 12
     |   ; CODE XREF from fcn.10001c46 (0x10001c88)
    .--> 0x10001c53      8b442408        mov eax, dword [arg_8h]     ; [0x8:4]=-1 ; 8
    :|   0x10001c57      8d0c06          lea ecx, [esi + eax]
    :|   0x10001c5a      0fb60406        movzx eax, byte [esi + eax]
    :|   0x10001c5e      c1e804          shr eax, 4
    :|   0x10001c61      83f809          cmp eax, 9                  ; 9
   ,===< 0x10001c64      7e04            jle 0x10001c6a
   |:|   0x10001c66      0437            add al, 0x37                ; '7'
  ,====< 0x10001c68      eb02            jmp 0x10001c6c
  ||:|   ; CODE XREF from fcn.10001c46 (0x10001c64)
  |`---> 0x10001c6a      0430            add al, 0x30                ; '0'
  | :|   ; CODE XREF from fcn.10001c46 (0x10001c68)
  `----> 0x10001c6c      8802            mov byte [edx], al
    :|   0x10001c6e      8a01            mov al, byte [ecx]
    :|   0x10001c70      83e00f          and eax, 0xf
    :|   0x10001c73      83f809          cmp eax, 9                  ; 9
   ,===< 0x10001c76      7e04            jle 0x10001c7c
   |:|   0x10001c78      0437            add al, 0x37                ; '7'
  ,====< 0x10001c7a      eb02            jmp 0x10001c7e
  ||:|   ; CODE XREF from fcn.10001c46 (0x10001c76)
  |`---> 0x10001c7c      0430            add al, 0x30                ; '0'
  | :|   ; CODE XREF from fcn.10001c46 (0x10001c7a)
  `----> 0x10001c7e      884201          mov byte [edx + 1], al
    :|   0x10001c81      46              inc esi
    :|   0x10001c82      42              inc edx
    :|   0x10001c83      42              inc edx
    :|   0x10001c84      3b742410        cmp esi, dword [arg_10h]    ; [0x10:4]=-1 ; 16
    `==< 0x10001c88      7cc9            jl 0x10001c53
     |   ; CODE XREF from fcn.10001c46 (0x10001c4d)
     `-> 0x10001c8a      5e              pop esi
         0x10001c8b      c20c00          ret 0xc
```

Explanation - Zignature Hashes

```
[0x10001c46]> zaf fcn.10001c46 BcdToAsc


[0x10001c46]> z


BcdToAsc:
  bytes: 5633f639......7e..8b......8b......8d....0fb6....c1e80483....7e..0
437eb..043088028a..83e00f83....7e..0437eb..043088420146424 23b......7c..5ec
20c00
  graph: cc=5 nbbs=10 edges=13 ebbs=1
  offset: 0x10001c46
```

Performance - Unscientific** Study

Input	Process Method	Process Time	Output	Match Time
msvcrNN.dll	Zignatures	Acceptable Time	Acceptable size	Acceptable time
mfcNN.dll	Zignatures	Long Time	Large (~60MB)	Hanus
Dlls from VC6 - VC14	Zignatures + Hashing	~1 hour	6MB	Near-instant
Dlls from WinXP System32	Zignatures + Hashing	2 hours	28MB	Near-instant

** This was done in February 2018. Changes in the core Zignatures code would affect these outcomes.

Implementation - Zignature Hashes

```
class ZigGenerator(Generator):

    sigtype = 'zighash'

    def __init__(self, r2):

        Generator.__init__(self, r2)


    def generate_hashes(self):

        hashes = {}
        temphashes = set()

        funcs = self.r2.cmdj('aflj')

        for func in funcs:
```

```
for func in funcs:

    # Generate the full zignature.
    self.r2.cmd('zaf ' + func['name'] + ' ' + func['name'])

    # View the zignature, get the bytes, and hash them.
    zigs = self.r2.cmdj('zj')

    # Hacky but only a single sig can ever exist at once.
    zig_bytes = zigs[0]['bytes']

    sig_byte_hash = hashlib.md5(zig_bytes.encode()).hexdigest()

    # Only create if the function is not too short and the
    # function hash is not stored already
    if len(zig_bytes) > 30 and sig_byte_hash not in temphashes:

        # Add the func+hash pair to the dict
        hashes[func['name']] = (
            sig_byte_hash
        )

        # Update set
        temphashes.add(sig_byte_hash)

    # Delete all zignatures to keep continuous
    # assurance only one exists at a time.
    self.r2.cmd('z-*')

return hashes
```

Explanation - String Set Hashes

```
Sqlite3_TCP.dll
lame_set_out_sample
ServiceMain
.?AVtype_info@@
\t\a\f\b\f\t\f\n\a\v\b\f

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
.?AVbad_exception@std@@
current_time
current_date
current_timestamp
%Y-%m-%d %H:%M:%S
922337203685477580
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
%s\\etilqs_
invalid page number %d
2nd reference to page %d
Failed to read ptrmap key=%d
Bad ptr map entry key=%d expected=(%d,%d) got=(%d,%d)
%d of %d pages missing from overflow list starting at %d
failed to get page %d
freelist leaf count too big on page %d
unable to get the page. error code=%d
sqlite3BtreeInitPage() returns error code %d
On tree page %d cell %d:
Child page depth differs
```

```
foreign_key_list
case_sensitive_like
integrity_check
integrity_check
*** in database %s ***\n
 missing from index
wrong # of entries in index
unsupported encoding: %s
schema_version
user_version
freelist_count
malformed database schema
sqlite_temp_master
sqlite_master
sqlite_temp_master
sqlite_master
```

Explanation - String Set Hashes

```
[0x10072e40]> pdsf @ 0x10049620
;-- r2kit_analyzed_func:
;-- library_func:
0x10049638 call fcn.100356b0
0x1004964d str.Failed_to_read_ptrmap_key__d
0x1004965a call fcn.10049540
0x1004968e str.Bad_ptr_map_entry_key__d_expected___d__d__got___d__d
0x1004969b call fcn.10049540
```

Implementation - String Set Hashes

```
class StringSetGenerator(Generator):

    sigtype = 'stringsethash'

    def __init__(self, r2):

        Generator.__init__(self, r2)


    def generate_hashes(self):

        hashes = {}
        temphashes = set()
        string_sets = {}

        strings = self.r2.cmdj("izzj")
```

```
# First, get the strings and for each string, make sure it is
# ascii or wide.
if strings:
    for string in strings['strings']:
        if string['type'] == 'ascii' or string['type'] == 'utf8':

            # Next, get the cross references to the string.
            xrefto = self.r2.cmdj("axtj " + str(string['vaddr']))
            if xrefto:
                for xref in xrefto:

                    # If the xref comes from a function, either add it
                    # to the list or add a new dictionary item.
                    if ('fcn_name' in xref and
                        len(base64.b64decode(string['string'])) >= 10):

                        if xref['fcn_name'] in string_sets:

                            string_sets[xref['fcn_name']].append(
                                base64.b64decode(string['string'])
                            )

                        elif xref['fcn_name'] not in string_sets:

                            string_sets[xref['fcn_name']] = (
                                [base64.b64decode(string['string'])]
                            )
```

Implementation - String Set Hashes

```
# After generating a dict of function name keys matched to referenced
# strings in a list as the value, hash the distinct values and handle
# collisions.
for func, stringset in string_sets.iteritems():

    stringsethash = hashlib.md5(''.join(stringset)).hexdigest()

    if stringsethash not in temphashes:

        temphashes.add(stringsethash)
        hashes[func] = (
            stringsethash
        )


return hashes
```

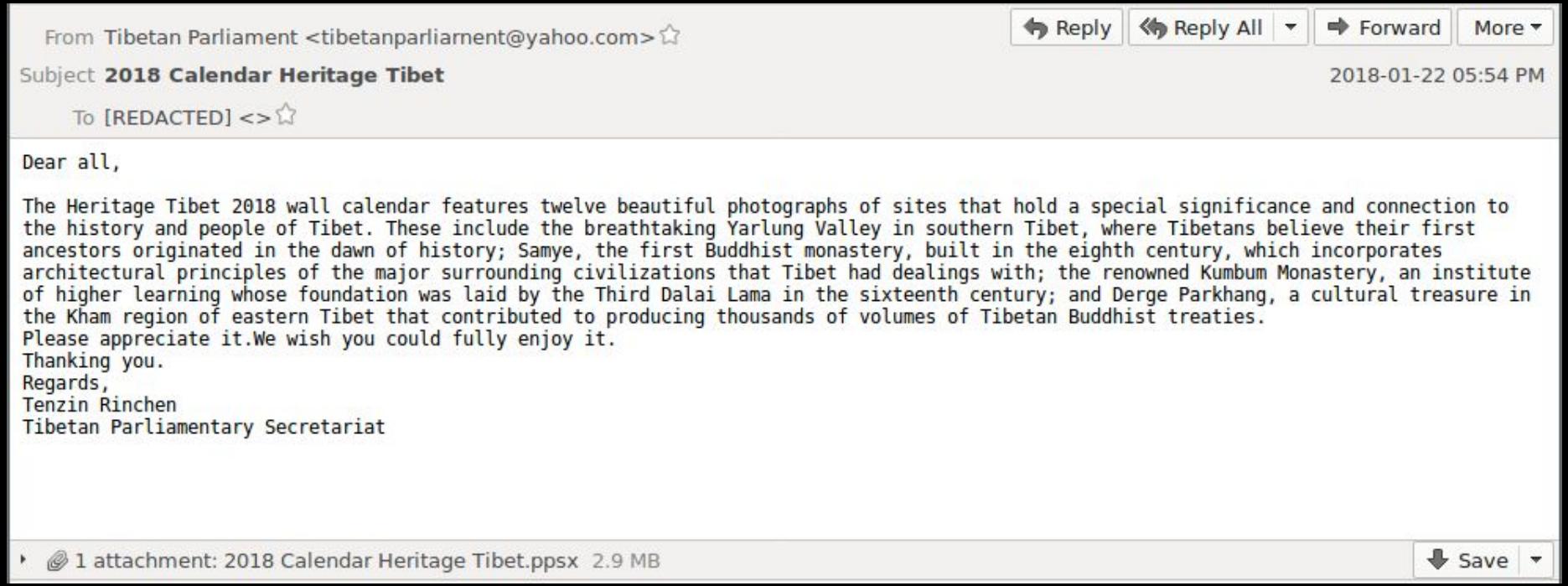
Explanation - Call Chain Hashes

```
[0x10003390]> pdsf @ 0x10001a7e
0x10001a8e str.Hardware__Description__System__CentralProcessor__0
0x10001a9b call dword [sym.imp.ADVAPI32.dll_RegOpenKeyExA]
0x10001ab6 str.MHz
0x10001abe call dword [sym.imp.ADVAPI32.dll_RegQueryValueExA]
0x10001ace call dword [sym.imp.ADVAPI32.dll_RegCloseKey]
```

```
GADGETS = [
{'name': 'enumerate_processes', 'api_chain': ['CreateToolhelp32Snapshot','Process32First','Process32Next']},
{'name': 'search_files', 'api_chain': ['FindFirstFile','FindNextFile','FindClose']},
{'name': 'query_regkey', 'api_chain': ['RegOpenKey','RegQueryValue','RegCloseKey']},
{'name': 'access_resources', 'api_chain': ['FindResource','LoadResource','LockResource']}
]
```

Application

Application - Zignature Hashes



Application - Zignature Hashes

```
$ md5sum psdll1
11e0f3e1c7d8855ed7f1dcfce4b7702a  psdll1
$ rabin2 -I psdll1 | grep --color=never compiled
compiled Mon Jan 15 10:58:08 2018
$ rabin2 -zzqq psdll1 | grep --color=never pdb
C:\\Users\\learn\\Desktop\\免杀\\work\\ConsoleApplication1\\Release\\dll.pdb
$ rabin2 -l psdll1
[Linked libraries]
kernel32.dll
msvcr110.dll

2 libraries
```

Application - Zignature Hashes

```
found PE optional header (size: 224)
        Magic Number: PE32
                Magic: 0x10b
        major linker version: 11
        minor linker version: 0
        size of code: 3072
        size of initialized data: 13824
        size of uninitialized data: 0
        code entry point: 5043 (execution starts here)
        base of code: 4096
        base of data: 8192
        image base: 268435456 (0x10000000)
                uncommon image base
        section alignment: 4096
        file alignment: 512
        MajorOperatingSystemVersion: 6
        MinorOperatingSystemVersion: 0
        MajorImageVersion: 0
        MinorImageVersion: 0
        MajorSubSystemVersion: 6 (0x6)
        MinorSubSystemVersion: 0 (0x0)
```

Application - Zignature Hashes

```
[0x100013b3]> #!pipe python ./sessionstarter.py -l sigs/vc32bit.zighash
[x] Analyze all flags starting with sym. and entry0 (aa)
Matching from sigs/vc32bit.zighash...
[0x10001a2c]> afll
address    size  nbbs edges    cc cost  min bound range max bound  calls locals args xref frame name
========== ==== ===== ===== ===== ==== ========== ===== ========== ===== ====== ==== ==== ===== ====
0x10001000  355    19    27    10  173 0x10001000   355 0x10001163     9     2       1    2    80 fcn.10001000
0x10001163  264     3     2     1    8 0x10001163    15 0x10001172     0     1       2    2   816 fcn.10001163
0x100011be  501    41    61    22  250 0x100011be   501 0x100013b3    13     2       3    3    24 fcn.100011be
0x100013b3   35     3     3     2   23 0x100013b3    35 0x100013d6     2     0       3    0    16 entry0
0x100013d6  268    20    33    15  157 0x100013d6   293 0x100014fb     8     2       3    1    96 vc11_mfc110.dll_fcn.1024a2a8
0x100014fe   11     1     0     1    4 0x100014fe    11 0x10001509     0     0       0    1     0 globalassign fcn100014fe
0x10001509   61     3     3     2   36 0x10001509    61 0x10001546     5     0       1    1    12 vc11_mfc110.dll_fcn.10249e2a
0x10001546  249     3     3     2   64 0x10001546   249 0x1000163f     2     1       2    1   816 fcn.10001546
```

Application - Zignature Hashes

```
$ md5sum psdll2
00c4f776ba6d8a6e2d31a212f25fc2cc  psdll2
$ rabin2 -I psdll2 | grep --color=never compiled
compiled Mon Jan 29 11:57:02 2018
$ rabin2 -zzqq psdll2 | grep --color=never pdb
c:\\Users\\learn1\\Documents\\Visual Studio 2005\\Projects\\psdll\\release\\psdll.pdb
$ rabin2 -l psdll2
[Linked libraries]
kernel32.dll
msvcr80.dll

2 libraries
```

Application - Zignature Hashes

```
found PE optional header (size: 224)
        Magic Number: PE32
                Magic: 0x10b
        major linker version: 8
        minor linker version: 0
        size of code: 2560
        size of initialized data: 12800
        size of uninitialized data: 0
        code entry point: 5247 (execution starts here)
        base of code: 4096
        base of data: 8192
        image base: 268435456 (0x10000000)
                uncommon image base
        section alignment: 4096
        file alignment: 512
        MajorOperatingSystemVersion: 4
        MinorOperatingSystemVersion: 0
        MajorImageVersion: 0
        MinorImageVersion: 0
        MajorSubSystemVersion: 4 (0x4)
        MinorSubSystemVersion: 0 (0x0)
```

Application - Zignature Hashes

```
[0x1000147f]> #!pipe python ./sessionstarter.py -l sigs/vc32bit.zighash
[x] Analyze all flags starting with sym. and entry0 (aa)
Matching from sigs/vc32bit.zighash...
[0x100018d6]> afll
address    size  nbbs edges    cc cost  min bound range max bound  calls locals args xref frame name
========== ==== ===== ===== ===== ==== ========== ===== ========== ===== ====== ==== ==== ===== ====
0x10001000  363    19    27    10  170 0x10001000   366 0x1000116e     7      5    1    2    76 fcn.10001000
0x1000116e  275     3     2     1    8 0x1000116e    15 0x1000117d     0      3    2    2   828 fcn.1000116e
0x100011ca  415    38    53    17  221 0x100011ca   415 0x10001369    12      1    1    3    64 fcn.100011ca
0x10001369  240    21    35    16  155 0x10001369   267 0x10001474     8      2    1    1   100 vc8_mfc80.dll_fcn.78295363
0x10001474   11     1     0     1    4 0x10001474    11 0x1000147f     0      0    0    1     0 globalassign_fcn10001474
0x1000147f   33     3     3     2   17 0x1000147f    33 0x100014a0     2      0    4    0     4 entry0
```

Application - String Set Hashes

-------- Forwarded Message --------
Subject: Fwd: 民主黨議員及高教界選委將全投或傾向投票給曾俊華.
Date: Tue, 21 Mar 2017 11:46:39 +0800

建源（收）

距離行政長官選舉投票只有幾天，「民主300+」選委今晚開會討論投票意向，民主黨立法會議員及高教界選委，分別宣布會全投或傾向投票給曾俊華。

對於「公民聯合行動」籌辦的特首民間投票計劃，投票人數遠低於100萬的目標。有建制派選委認為，投票計劃失敗，反映市民不滿和不信任此類網上投票。

W 公民聯合行動 民主300+.doc

2 Attachments

Application - String Set Hashes

```
[0x10072e40]> #!pipe python /home/cmb/dev/r2kit/sessionstarter.py -l /home/cmb/dev/r2kit/sigs/sqlite3.stringsethash
Matching from /home/cmb/dev/r2kit/sigs/sqlite3.stringsethash...

[0x10072e40]> afll ~sqlite
0x1002e8a0   68    6    7    3   38 0x1002e8a0   68 0x1002e8e4    2    0    2    2    8 sqlite3_3071600.dll_fcn.609145ac
0x1003b550   52    3    3    2   28 0x1003b550   52 0x1003b584    1    1    1    2   20 sqlite3_3071600.dll_fcn.6090259c
0x10043f50  546   24   35   13  221 0x10043f50  546 0x10044172    6    4    1    1   36 sqlite3_3071600.dll_fcn.60935121
0x10044180  292    6    7    3  130 0x10044180  292 0x100442a4    7    3    1    1   28 sqlite3_3071600.dll_fcn.60935121
0x10049620  135    6    8    4   81 0x10049620  135 0x100496a7    3    3    5    7   48 sqlite3_3071700.dll_fcn.6092adf6
0x100496b0  542   27   38   13  285 0x100496b0  542 0x100498ce   15    7    5    2   48 sqlite3_3071600.dll_fcn.60929dd3
0x100498d0  185   12   16    6   88 0x100498d0  185 0x10049989    2    1    3    3   28 sqlite3_3080300.dll_fcn.60918df3
0x10053ad0  993   54   81   29  398 0x10053ad0  993 0x10053eb1   16    8    3    4   44 sqlite3_3071600.dll_fcn.60955e3f
0x10056b20  220    8   10    4  112 0x10056b20  220 0x10056bfc    6    3    2    1   36 sqlite3_3110000.dll_fcn.61e669dd
0x100570f0   42    4    5    3   26 0x100570f0   42 0x1005711a    1    0    4    3    8 sqlite3_3071600.dll_fcn.6091976c
0x100577a0  145    6    7    3   86 0x100577a0  145 0x10057831    5    3    3    2   40 sqlite3_3071600.dll_fcn.60917007
0x1005abe0  260   12   16    6  101 0x1005abe0  260 0x1005ace4    2    1    2   12   20 sqlite3_3071600.dll_fcn.609176e3
0x1005c610   98    7   10    5   51 0x1005c610   98 0x1005c672    2    0    2    3    8 sqlite3_3071600.dll_fcn.609175c1
0x1005ce70  189    6    8    4   96 0x1005ce70  189 0x1005cf2d    6    4    2    4   24 sqlite3_3071600.dll_fcn.60918d3d
0x1005cf60  476   32   46   16  208 0x1005cf60  476 0x1005d13c    6    4    5    2   52 sqlite3_3071600.dll_fcn.60957b08
0x1005e120  931   43   61   20  382 0x1005e120  933 0x1005e4c5   12    9    5    2   44 sqlite3_3071600.dll_fcn.60918dd8
0x1005f170  189   12   15    5   96 0x1005f170  189 0x1005f22d    3    3    2    1   28 sqlite3_3071600.dll_fcn.6091909b
0x1005fe70  455   24   35   13  253 0x1005fe70  460 0x1006003c   17    9    3    2   52 sqlite3_3071600.dll_fcn.60950f84
0x10060500  111    7    9    4   58 0x10060500  111 0x1006056f    2    2    2    1   20 sqlite3_3071600.dll_fcn.6091c65b
0x10060880 1867   76  114   40  964 0x10060880 1867 0x10060fcb   57   19    3    1  128 sqlite3_3071600.dll_fcn.6094c2bc
0x10061450 3893  195  288   95 1930 0x10061450 3893 0x10062385  105   33    6    3  184 sqlite3_3080200.dll_fcn.60951618
0x10066590  110    6    7    3   55 0x10066590  110 0x100665fe    3    1    1    2    8 sqlite3_3071600.dll_fcn.6093708c
0x100680d0  446   26   36   12  191 0x100680d0  446 0x1006828e    5   11    5    2   64 sqlite3_3071600.dll_fcn.60917a4e
0x100684f0 3754  146  210   66 1753 0x100684f0 3754 0x1006939a   93   55    7   15  292 sqlite3_3081000.dll_fcn.61c1c5b9
0x1006c310 3520  175  250   77 1683 0x1006c310 3520 0x1006d0d0   84   27    5    1  160 sqlite3_3071600.dll_fcn.6094a58a
0x10070e20 1090   57   79   24  443 0x10070e20 1090 0x10071262   11   17    8    1   88 sqlite3_3120200.dll_fcn.61e29efa
```

Application - String Set Hashes

```
[0x10072e40]> pdsf @ 0x10049620
;-- r2kit_analyzed_func:
;-- library_func:
0x10049638 call fcn.100356b0
0x1004964d str.Failed_to_read_ptrmap_key__d
0x1004965a call fcn.10049540
0x1004968e str.Bad_ptr_map_entry_key__d_expected___d__d__got___d__d
0x1004969b call fcn.10049540
```

<<-- From malware payload

<<-- From sqlite3 library

```
[0x60901058]> pdsf @ 0x6092adf6
0x6092ae14 call sub.database_corruption_at_line__d_of___.10s_c44
0x6092ae34 str.Failed_to_read_ptrmap_key__d
0x6092ae46 call fcn.60915a9c
0x6092ae78 str.Bad_ptr_map_entry_key__d_expected___d__d__got___d__d
0x6092ae8a call fcn.60915a9c
```

Future Work

- Improved naming for non-export DLL functions
- Expand Zighash sets
- Study frequently occurring function-level call chains
- Spend time on edge cases
- Repackage outside r2kit (with additional outputs)
- ….and python3, of course

El fin

Code: https://github.com/cmatthewbrooks/r2kit