txOne networks *The Leader of OT Zero Trust*

反組譯建立次世代語意感知特徵碼引擎

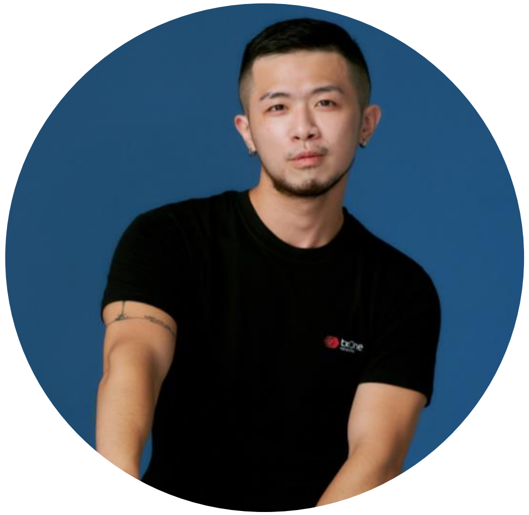
Sheng-Hao Ma

@aaaddress1

Hank Chen

@hank0438

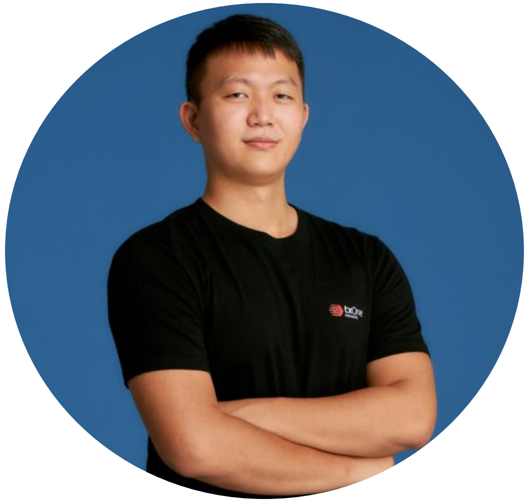
Who Are We?



Sheng-Hao Ma

Threat Researcher
PSIRT and Threat Research

- Spoke at Black Hat, DEFCON, HITB, VXCON, HITCON, ROOTCON, and CYBERSEC
- Instructor of CCoE Taiwan, Ministry of National Defense, Ministry of Education, and etc.
- The author of the popular security book "Windows APT Warfare: The Definitive Guide for Malware Researchers"



Hank Chen

Threat Researcher
PSIRT and Threat Research

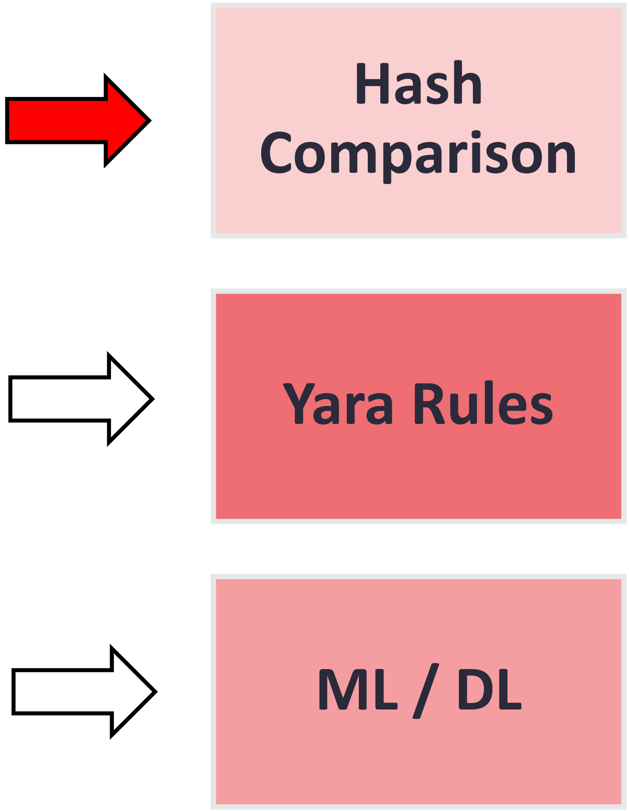
- Spoke at BlackHat USA, FIRST, HITCON, VXCON, and ThreatCon
- Instructor of Ministry of National Defense
- Teaching assistant of Cryptography and Information Security Course in Taiwan NTHU and CCoE Taiwan
- Member of CTF team 10sec and ⚔TSJ⚔

Outline

- Traditional Static Malware Analysis
- Static Malware Analysis in the Next Generation
- Conclusion

Traditional Static Malware Analysis

The Evolution of Static Malware Detection



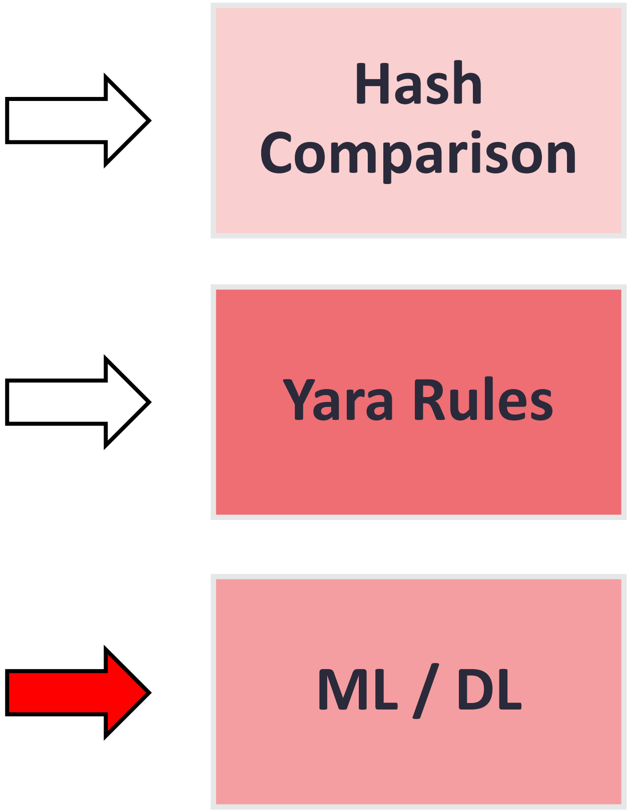
Hash Comparison

Yara Rules

ML / DL

- For Hash Comparison, analyst adopt fuzzy hash to identify the similarity of malwares
  - TLSH - A Locality Sensitive Hash

The Evolution of Static Malware Detection



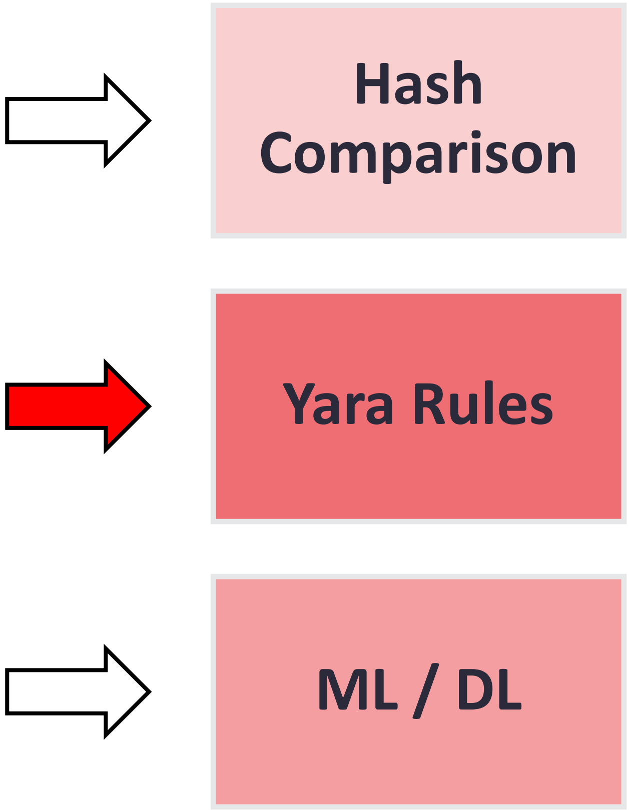
Hash Comparison

Yara Rules

ML / DL

- For ML / DL, there are lots of research based on machine learning or neural network to classify malware families
  - SVM, random forest, Ngram, asm2vec

The Evolution of Static Malware Detection

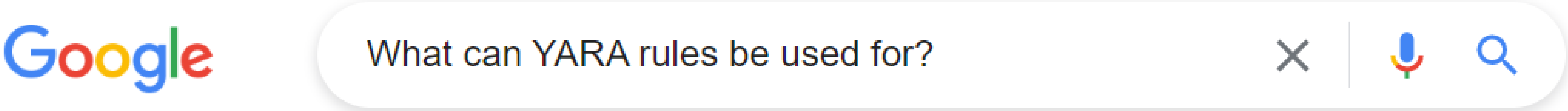


Hash Comparison

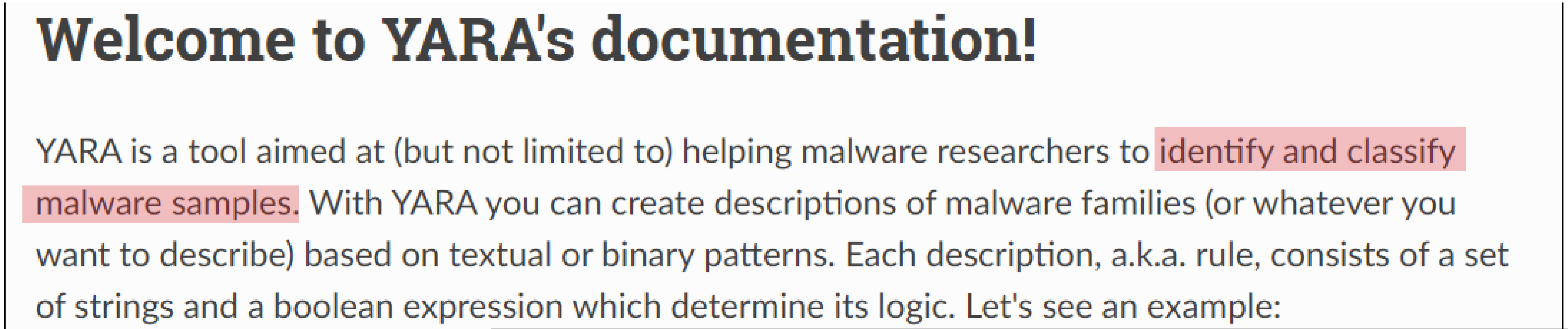
Yara Rules

ML / DL

- For Yara rules, analyst make some rules for the strings / byte sequences fetched from the binary



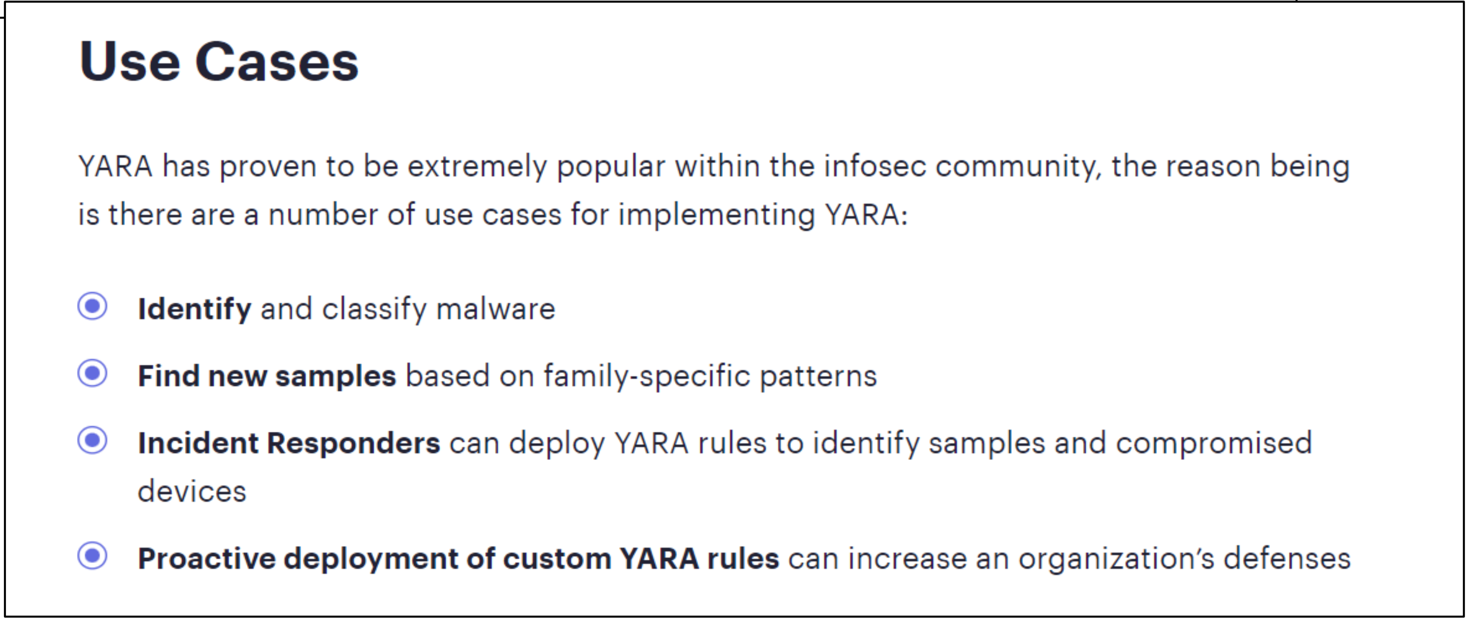
What can YARA rules be used for?



Welcome to YARA's documentation!

YARA is a tool aimed at (but not limited to) helping malware researchers to identify and classify malware samples. With YARA you can create descriptions of malware families (or whatever you want to describe) based on textual or binary patterns. Each description, a.k.a. rule, consists of a set of strings and a boolean expression which determine its logic. Let's see an example:

https://yara.readthedocs.io/en/stable/



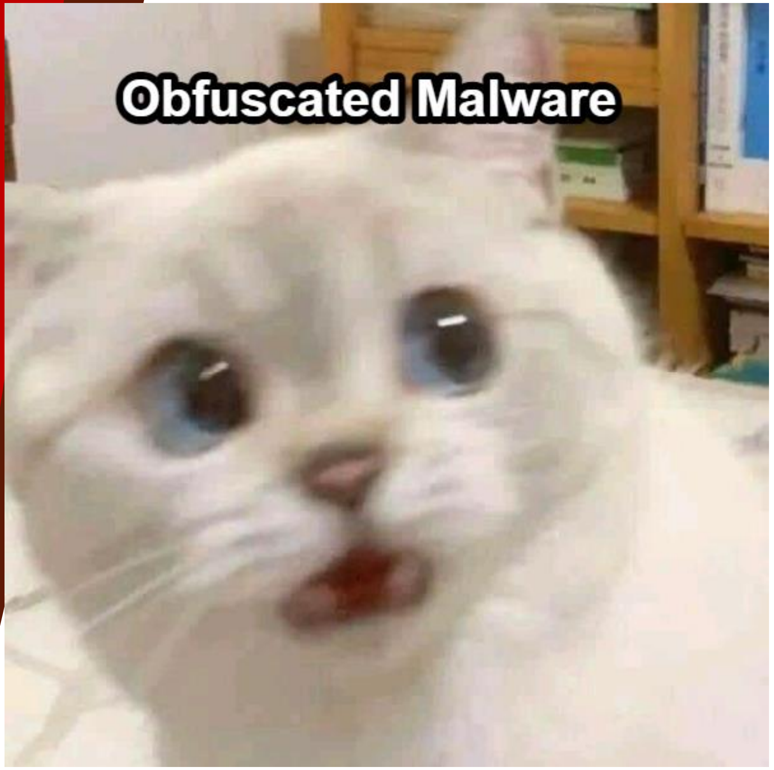
Use Cases

YARA has proven to be extremely popular within the infosec community, the reason being is there are a number of use cases for implementing YARA:

- **Identify** and classify malware
- **Find new samples** based on family-specific patterns
- **Incident Responders** can deploy YARA rules to identify samples and compromised devices
- **Proactive deployment of custom YARA rules** can increase an organization’s defenses

https://www.varonis.com/blog/yara-rules

Yara Rule of WannaCry

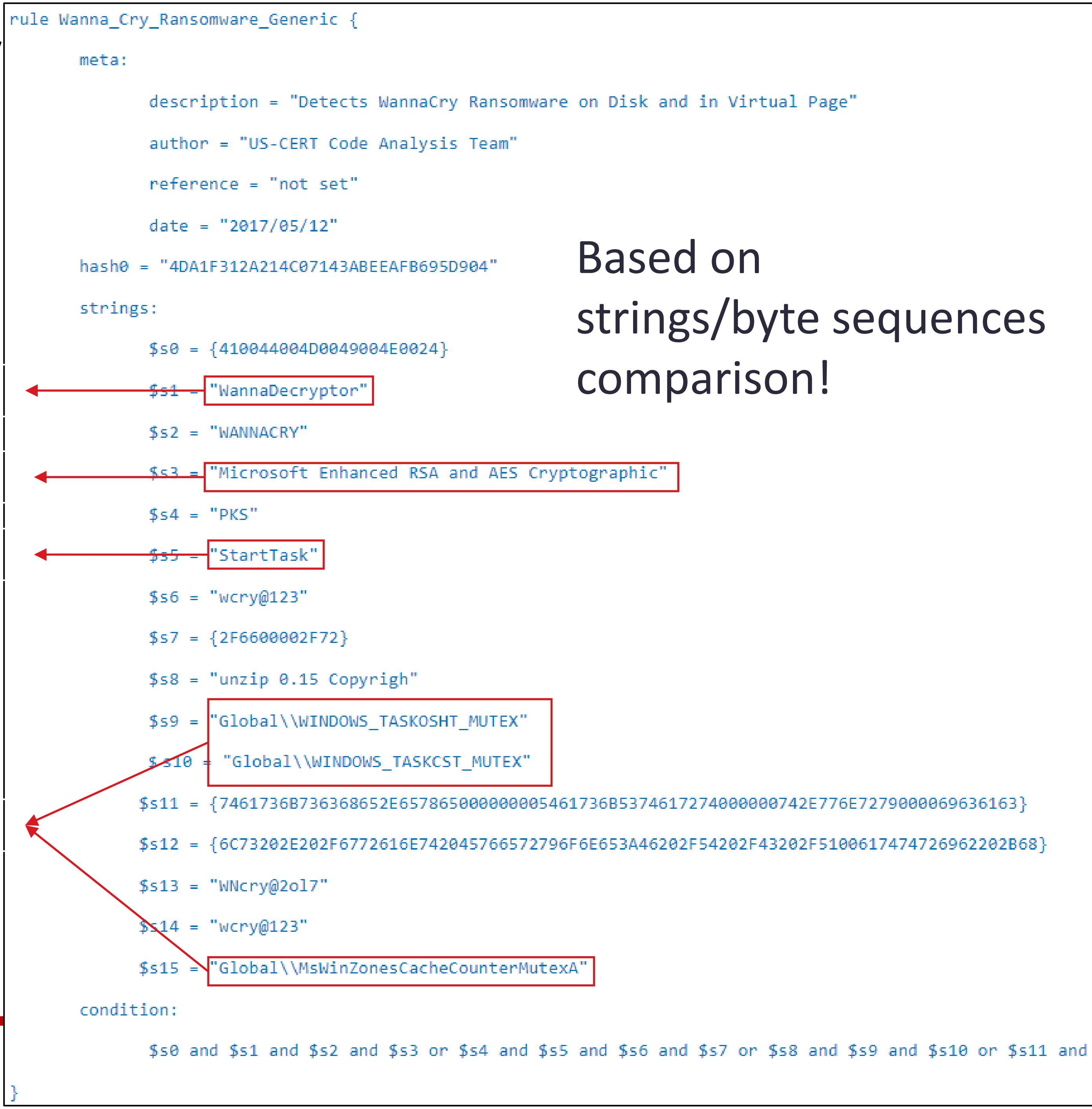


Drop file

Encryption

Windows API

Mutex



```
rule Wanna_Cry_Ransomware_Generic {
    meta:
        description = "Detects WannaCry Ransomware on Disk and in Virtual Page"
        author = "US-CERT Code Analysis Team"
        reference = "not set"
        date = "2017/05/12"
    hash0 = "4DA1F312A214C07143ABEEAFB695D904"
    strings:
        $s0 = {410044004D0049004E0024}
        $s1 = "WannaDecryptor"
        $s2 = "WANNACRY"
        $s3 = "Microsoft Enhanced RSA and AES Cryptographic"
        $s4 = "PKS"
        $s5 = "StartTask"
        $s6 = "wcry@123"
        $s7 = {2F6600002F72}
        $s8 = "unzip 0.15 Copyrigh"
        $s9 = "Global\\WINDOWS_TASKOSHT_MUTEX"
         $s10 = "Global\\WINDOWS_TASKCST_MUTEX"
       $s11 = {7461736B736368652E657865000000005461736B5374617274000000742E776E7279000069636163}
       $s12 = {6C73202E202F6772616E742045766572796F6E653A46202F54202F43202F5100617474726962202B68}
       $s13 = "WNcry@2ol7"
       $s14 = "wcry@123"
       $s15 = "Global\\MsWinZonesCacheCounterMutexA"
    condition:
        $s0 and $s1 and $s2 and $s3 or $s4 and $s5 and $s6 and $s7 or $s8 and $s9 and $s10 or $s11 and
}
```

Based on strings/byte sequences comparison!

Static Malware Analysis in The Next Generation

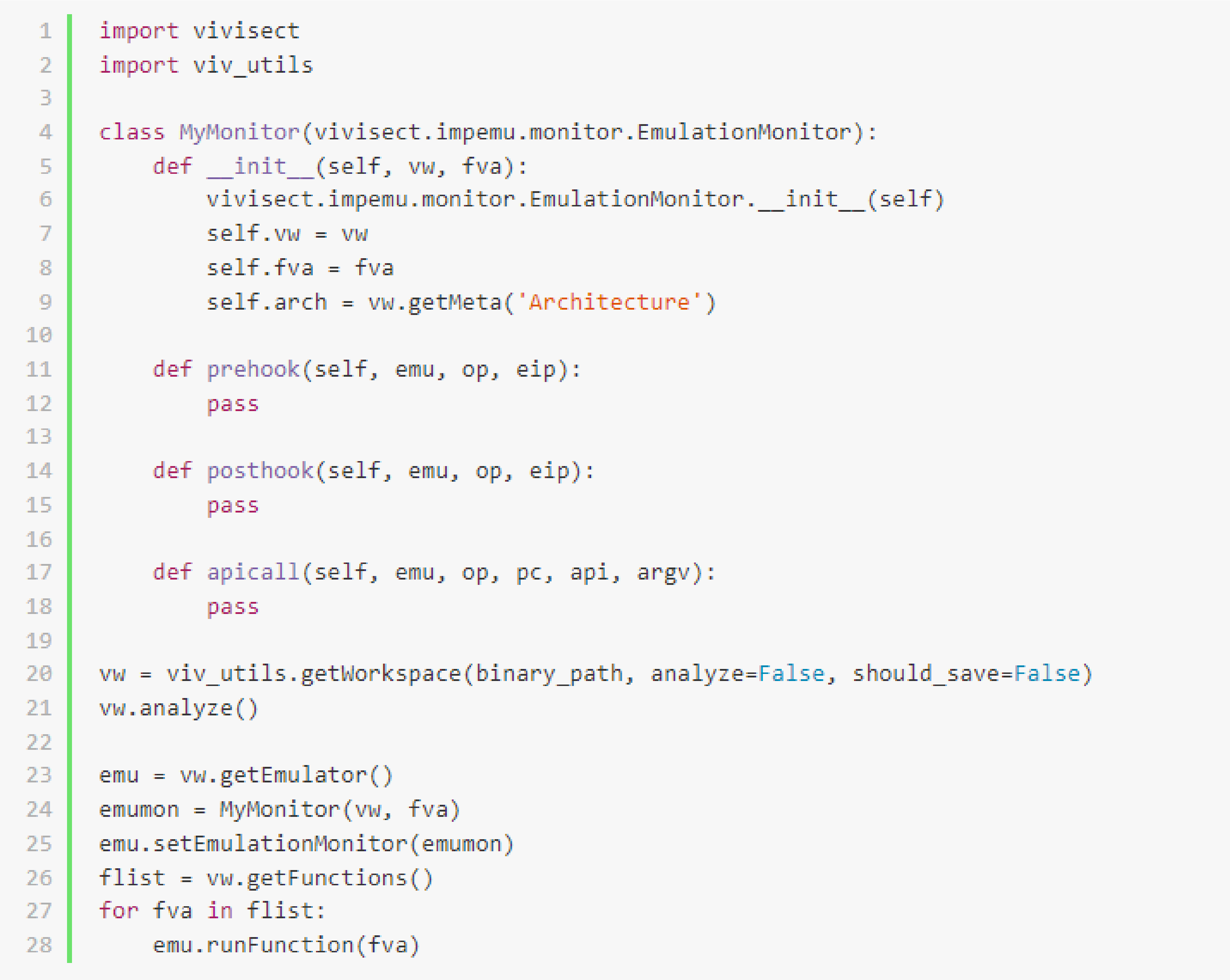
Static Malware Analysis in The Next Generation

- Vivisect
  - A combined disassembler/static analysis/symbolic execution/debugger framework
- Capa
  - Detect capabilities in executable files
- Flare-floss
  - Automatically deobfuscate strings from malware binaries

Vivisect

Vivisect

- A simple & lightweight static symbolic execution framework which help malware analyst to capture the signature of the binary in the execution time
  - Disassemble instructions
  - Reconstruct function
  - Rebuild CFG (cross references)
  - Emulation



```python
import vivisect
import viv_utils

class MyMonitor(vivisect.impemu.monitor.EmulationMonitor):
    def __init__(self, vw, fva):
        vivisect.impemu.monitor.EmulationMonitor.__init__(self)
        self.vw = vw
        self.fva = fva
        self.arch = vw.getMeta('Architecture')

    def prehook(self, emu, op, eip):
        pass

    def posthook(self, emu, op, eip):
        pass

    def apicall(self, emu, op, pc, api, argv):
        pass

vw = viv_utils.getWorkspace(binary_path, analyze=False, should_save=False)
vw.analyze()

emu = vw.getEmulator()
emumon = MyMonitor(vw, fva)
emu.setEmulationMonitor(emumon)
flist = vw.getFunctions()
for fva in flist:
    emu.runFunction(fva)
```

SoK: All You Ever Wanted to Know About Binary Disassembly But Were Afraid to Ask

- Findings
  - Heuristics are used to handle complex constructs which are common in binaries
  - Heuristics inherently introduce coverage-correctness trade-offs
- My criteria
  - Friendly user interfaces (Programming Languages, APIs, …)
  - High performance
  - Supportability
  - Correctness

Static Malware Analysis in The Next Generation

- Vivisect
  - A combined disassembler/static analysis/symbolic execution/debugger framework
- Capa
  - Detect capabilities in executable files
- Flare-floss
  - Automatically deobfuscate strings from malware binaries

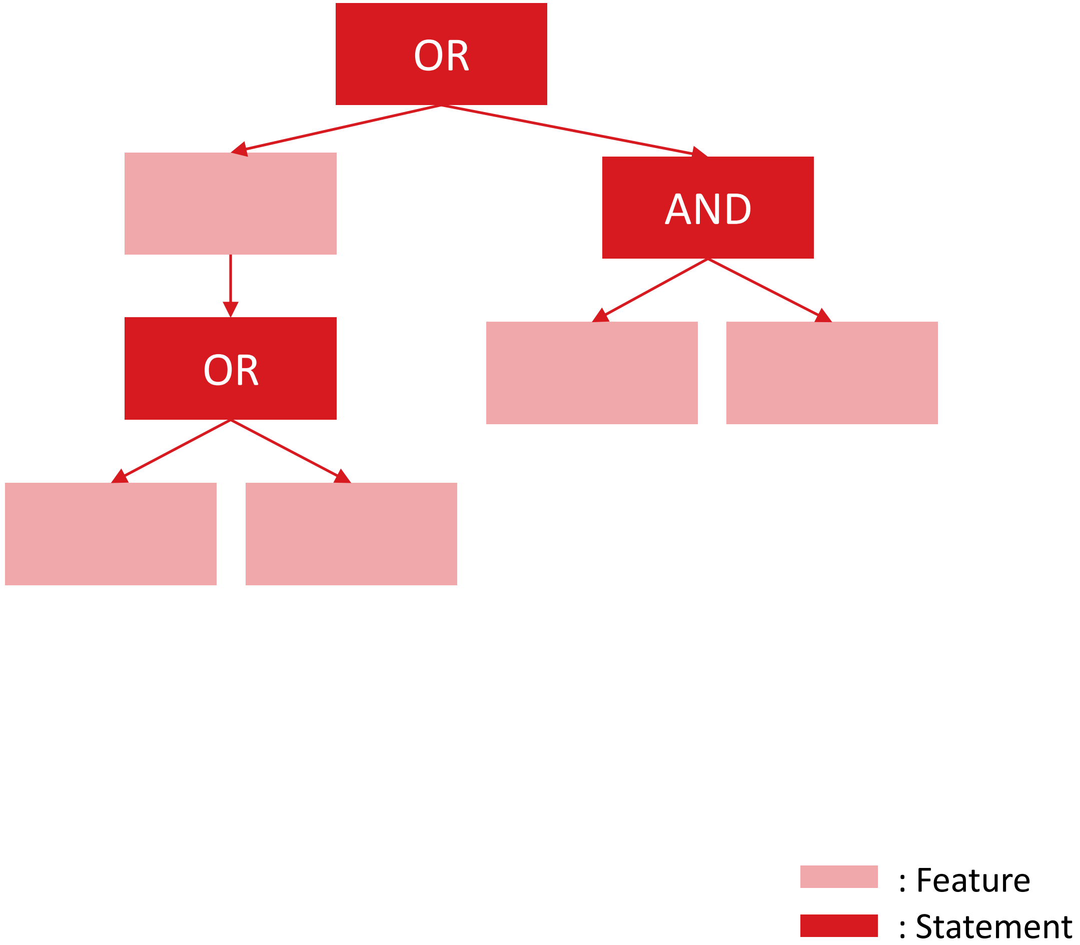
Capa

Capa

- A tool based on ***vivisect*** to extract features from instructions / basic blocks / functions in the binary
- Contains a variety of rules for malware analysis to detect malicious behaviors

Capa-rules

- Scope
  - File, Function, Basic Block, Instruction
- Node (AST)
  - Statement (Logical Expression)
    - and, or, optional, basic block, …
  - Feature
    - Import
    - String
    - Number
    - Bytes
    - Count
    - Match
    - …



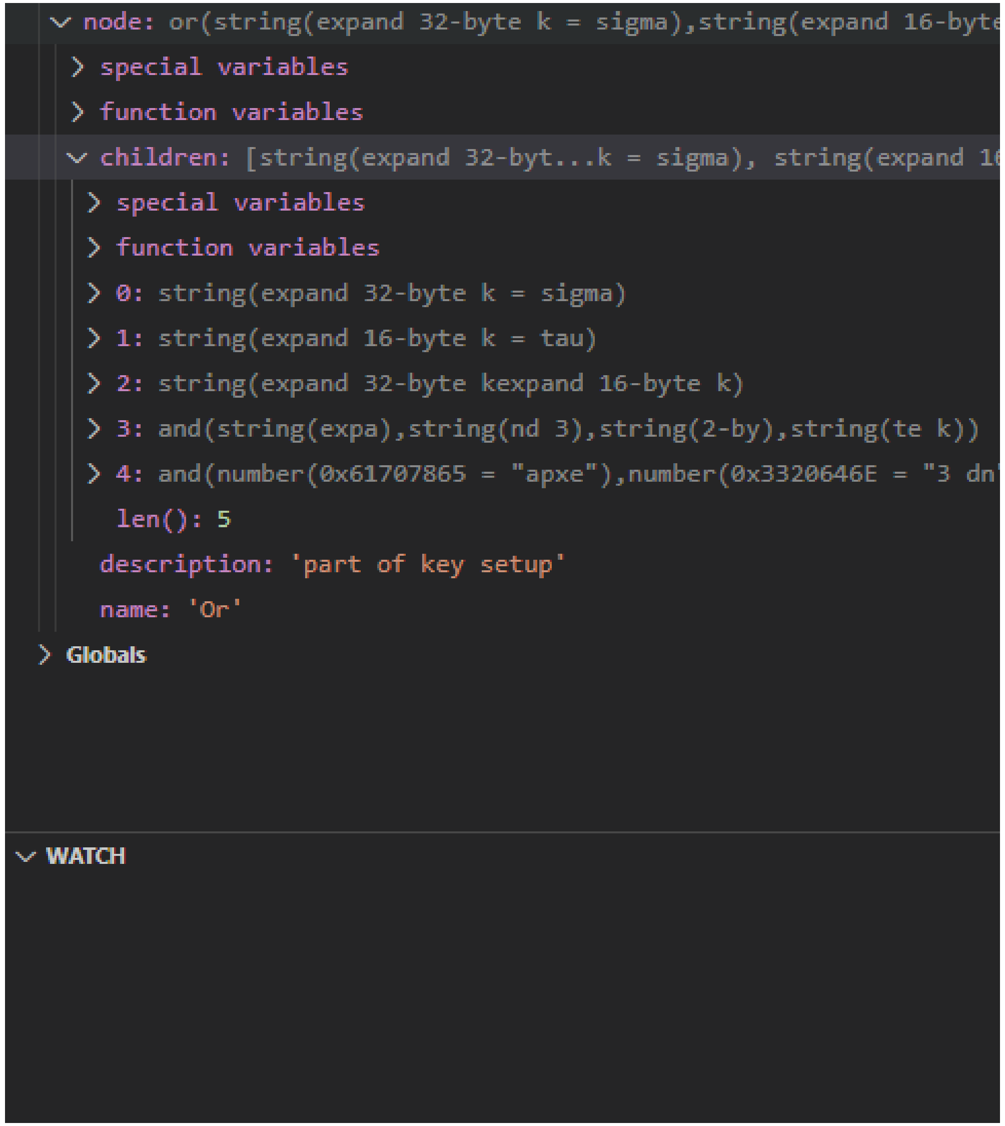
feature

feature

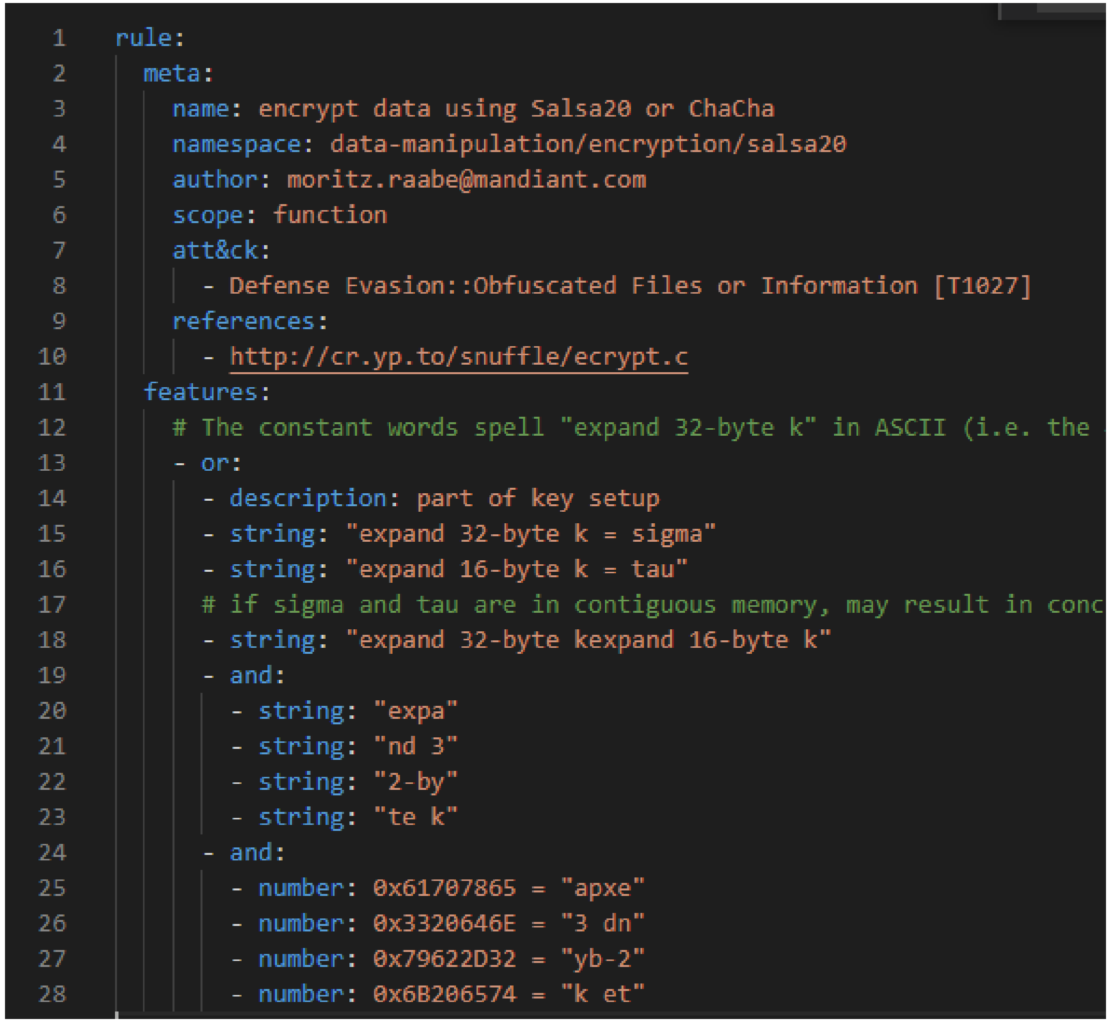
feature

statement

statement



```
node: or(string(expand 32-byte k = sigma),string(expand 16-byte
  special variables
  function variables
  children: [string(expand 32-byt...k = sigma), string(expand 16
    special variables
    function variables
    0: string(expand 32-byte k = sigma)
    1: string(expand 16-byte k = tau)
    2: string(expand 32-byte kexpand 16-byte k)
    3: and(string(expa),string(nd 3),string(2-by),string(te k))
    4: and(number(0x61707865 = "apxe"),number(0x3320646E = "3 dn'
    len(): 5
  description: 'part of key setup'
  name: 'Or'
Globals
```

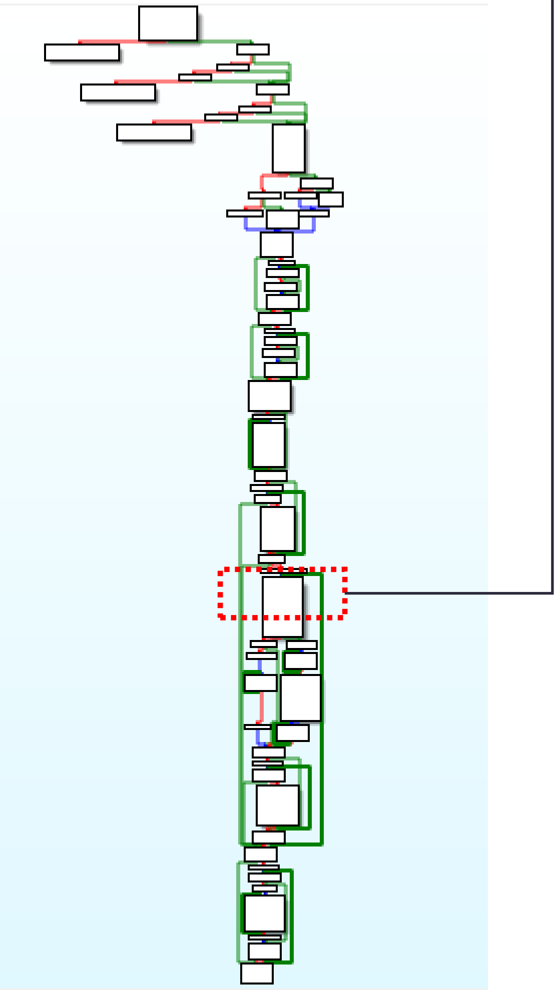


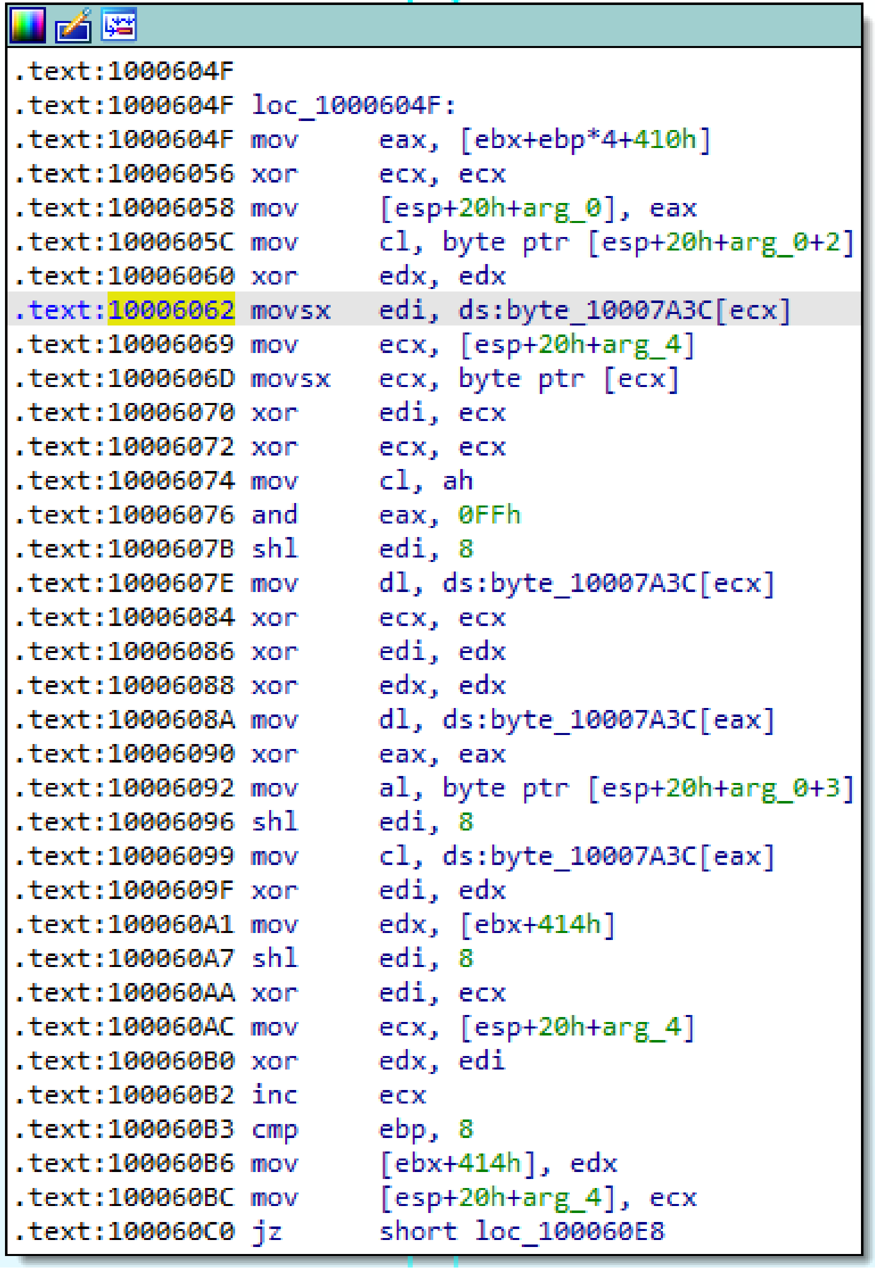
```yaml
rule:
  meta:
    name: encrypt data using Salsa20 or ChaCha
    namespace: data-manipulation/encryption/salsa20
    author: moritz.raabe@mandiant.com
    scope: function
    att&ck:
      - Defense Evasion::Obfuscated Files or Information [T1027]
    references:
      - http://cr.yp.to/snuffle/ecrypt.c
  features:
    # The constant words spell "expand 32-byte k" in ASCII (i.e. the
    - or:
      - description: part of key setup
      - string: "expand 32-byte k = sigma"
      - string: "expand 16-byte k = tau"
      # if sigma and tau are in contiguous memory, may result in conc
      - string: "expand 32-byte kexpand 16-byte k"
      - and:
        - string: "expa"
        - string: "nd 3"
        - string: "2-by"
        - string: "te k"
      - and:
        - number: 0x61707865 = "apxe"
        - number: 0x3320646E = "3 dn"
        - number: 0x79622D32 = "yb-2"
        - number: 0x6B206574 = "k et"
```

Case Study: Capa Rule with Ransomware

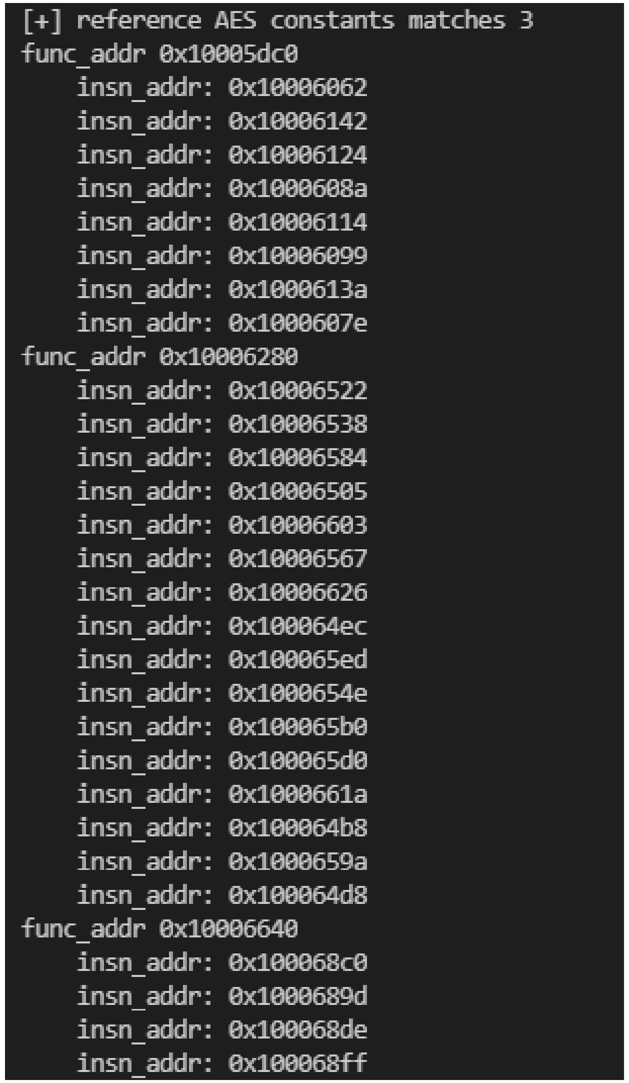
Malware	Sha256	Encryption Algorithm Catagories	Encryption File Function Address	Matched Capa Rule
WannaCry	4827723539f683fc8038a95d2fa2d8021401f136d28fa57f34d32c7cd23543ed	AES	0x10005dc0 0x10006280 0x10006640	reference AES constants
Conti v2	d3c75c5bc4ae087d547bd722bd84478ee6baf8c3355b930f26cc19777cd39d4c	Salsa20 / Chacha	0x405ac0 0x4105a0	encrypt data using Salsa20 or ChaCha
Conti v3 (exe)	E1B147AA2EFA6849743F570A3ACA8390FAF4B90AED490A5682816DD9EF10E473	Salsa20 / Chacha	0x405740 0x40efa0 0x41acf0	encrypt data using Salsa20 or ChaCha
Conti v3 (dll)	FB737DA1B74E8C84E6D8BD7F2D879603C27790E290C04A21E00FBDE5ED86EEE3	Salsa20 / Chacha	0x100056f0 0x1000ef70 0x1001acd0	encrypt data using Salsa20 or ChaCha
Lockbit 1.0	0a937d4fe8aa6cb947b95841c490d73e452a3cafcd92645afc353006786aba76	AES	0x409550 0x41cb10	encrypt data using AES via x86 extensions
Lockbit 2.0	0545f842ca2eb77bcac0fd17d6d0a8c607d7dbc8669709f3096e5c1828e1c049	AES	0x43d8b0 0x43d970	encrypt data using AES via x86 extensions
Locky	03f6ab1b482eac4acfb793c3e8d0656d7c33cddb5fc38416019d526f43577761	AES	0x4014e5	encrypt or decrypt via WinCrypt
GandCrab 4.1	f5e74d939a5b329dddc94b75bd770d11c8f9cc3a640dccd8dff765b6997809f2	Salsa20 / Chacha	0x403971	encrypt data using Salsa20 or ChaCha
Maze	dee863ffa251717b8e56a96e2f9f0b41b09897d3c7cb2e8159fcb0ac0783611b	Salsa20 / Chacha	0x41a850	encrypt data using Salsa20 or ChaCha
Babuk	1c022007b7babd03c59ff6029b4dcc23cd66039515dc445729cf55071699aa74	HC-128	0x40fe80	encrypt data using HC-128
Cerber	e8c6741d3d21068535fb6bb7fe676ecaa74eee06a655c7aa915fc39c0ee7ee16	AES	0x404be4	encrypt or decrypt via WinCrypt

WannaCry





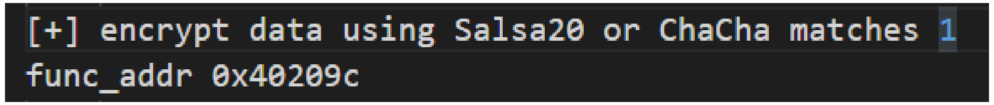
```
.text:1000604F
.text:1000604F loc_1000604F:
.text:1000604F mov     eax, [ebx+ebp*4+410h]
.text:10006056 xor     ecx, ecx
.text:10006058 mov     [esp+20h+arg_0], eax
.text:1000605C mov     cl, byte ptr [esp+20h+arg_0+2]
.text:10006060 xor     edx, edx
.text:10006062 movsx   edi, ds:byte_10007A3C[ecx]
.text:10006069 mov     ecx, [esp+20h+arg_4]
.text:1000606D movsx   ecx, byte ptr [ecx]
.text:10006070 xor     edi, ecx
.text:10006072 xor     ecx, ecx
.text:10006074 mov     cl, ah
.text:10006076 and     eax, 0FFh
.text:1000607B shl     edi, 8
.text:1000607E mov     dl, ds:byte_10007A3C[ecx]
.text:10006084 xor     ecx, ecx
.text:10006086 xor     edi, edx
.text:10006088 xor     edx, edx
.text:1000608A mov     dl, ds:byte_10007A3C[eax]
.text:10006090 xor     eax, eax
.text:10006092 mov     al, byte ptr [esp+20h+arg_0+3]
.text:10006096 shl     edi, 8
.text:10006099 mov     cl, ds:byte_10007A3C[eax]
.text:1000609F xor     edi, edx
.text:100060A1 mov     edx, [ebx+414h]
.text:100060A7 shl     edi, 8
.text:100060AA xor     edi, ecx
.text:100060AC mov     ecx, [esp+20h+arg_4]
.text:100060B0 xor     edx, edi
.text:100060B2 inc     ecx
.text:100060B3 cmp     ebp, 8
.text:100060B6 mov     [ebx+414h], edx
.text:100060BC mov     [esp+20h+arg_4], ecx
.text:100060C0 jz      short loc_100060E8
```



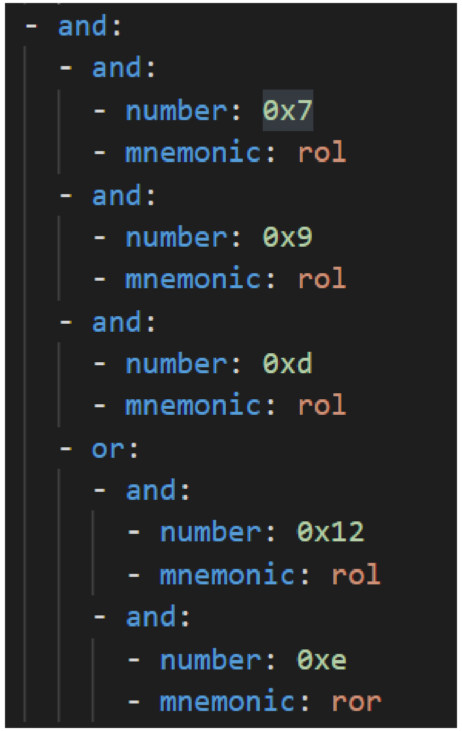
```
[+] reference AES constants matches 3
func_addr 0x10005dc0
     insn_addr: 0x10006062
     insn_addr: 0x10006142
     insn_addr: 0x10006124
     insn_addr: 0x1000608a
     insn_addr: 0x10006114
     insn_addr: 0x10006099
     insn_addr: 0x1000613a
     insn_addr: 0x1000607e
func_addr 0x10006280
     insn_addr: 0x10006522
     insn_addr: 0x10006538
     insn_addr: 0x10006584
     insn_addr: 0x10006505
     insn_addr: 0x10006603
     insn_addr: 0x10006567
     insn_addr: 0x10006626
     insn_addr: 0x100064ec
     insn_addr: 0x100065ed
     insn_addr: 0x1000654e
     insn_addr: 0x100065b0
     insn_addr: 0x100065d0
     insn_addr: 0x1000661a
     insn_addr: 0x100064b8
     insn_addr: 0x1000659a
     insn_addr: 0x100064d8
func_addr 0x10006640
     insn_addr: 0x100068c0
     insn_addr: 0x1000689d
     insn_addr: 0x100068de
     insn_addr: 0x100068ff
```

Darkside

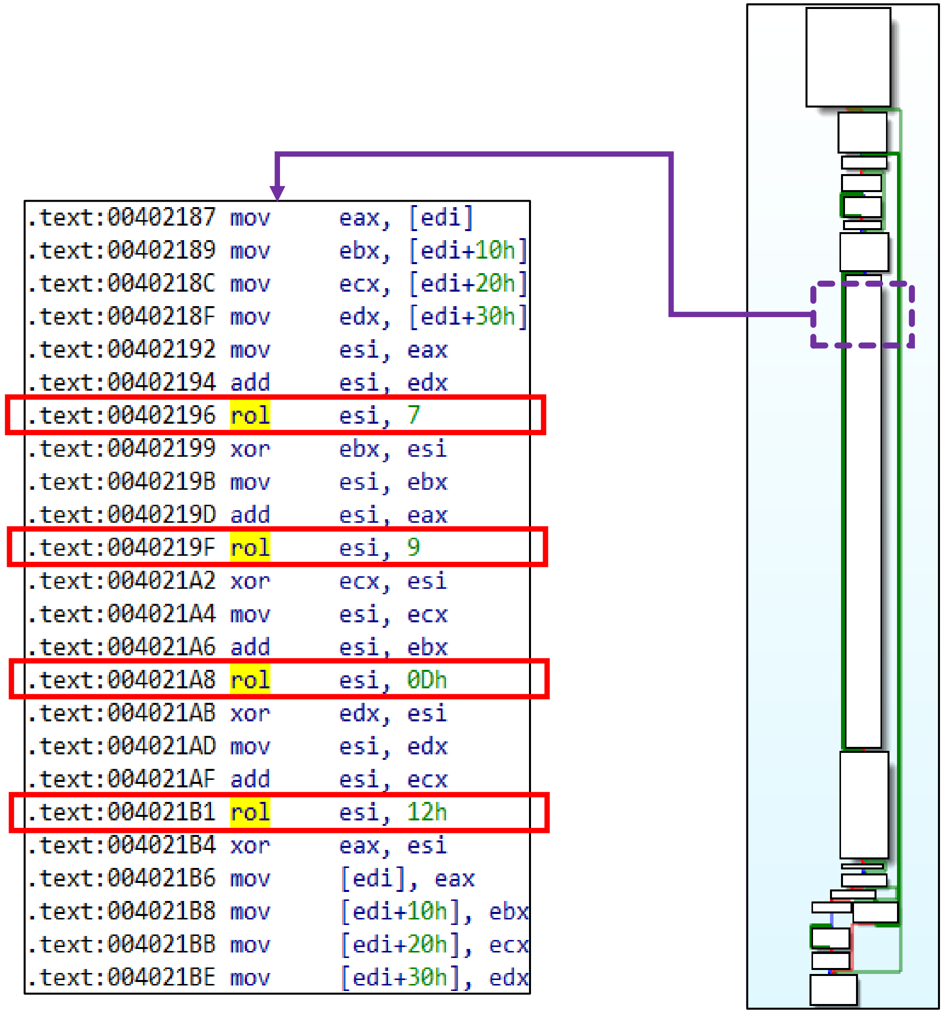
- Customized Salsa20 matrix and encryption
- 4 rounds of linear shifting



```
[+] encrypt data using Salsa20 or ChaCha matches 1
func_addr 0x40209c
```

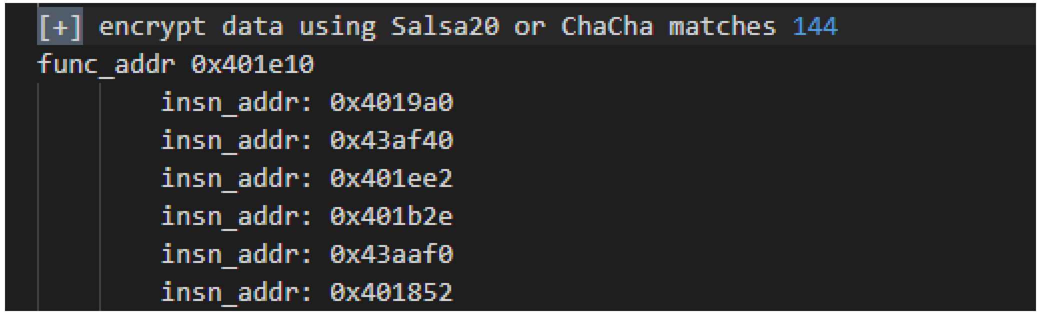


```
- and:
  - and:
    - number: 0x7
    - mnemonic: rol
  - and:
    - number: 0x9
    - mnemonic: rol
  - and:
    - number: 0xd
    - mnemonic: rol
  - or:
    - and:
      - number: 0x12
      - mnemonic: rol
    - and:
      - number: 0xe
      - mnemonic: ror
```

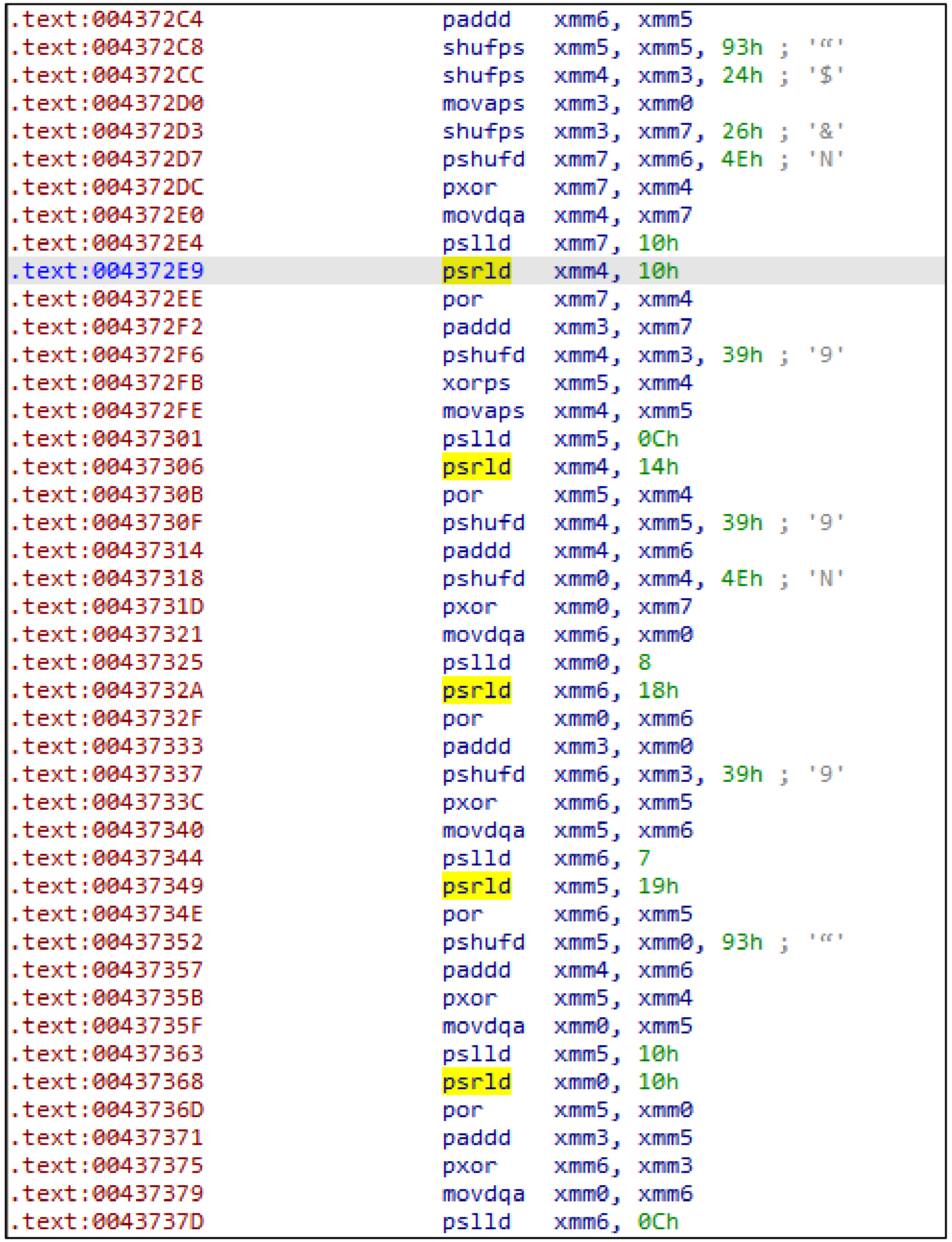


```
.text:00402187 mov     eax, [edi]
.text:00402189 mov     ebx, [edi+10h]
.text:0040218C mov     ecx, [edi+20h]
.text:0040218F mov     edx, [edi+30h]
.text:00402192 mov     esi, eax
.text:00402194 add     esi, edx
.text:00402196 rol     esi, 7
.text:00402199 xor     ebx, esi
.text:0040219B mov     esi, ebx
.text:0040219D add     esi, eax
.text:0040219F rol     esi, 9
.text:004021A2 xor     ecx, esi
.text:004021A4 mov     esi, ecx
.text:004021A6 add     esi, ebx
.text:004021A8 rol     esi, 0Dh
.text:004021AB xor     edx, esi
.text:004021AD mov     esi, edx
.text:004021AF add     esi, ecx
.text:004021B1 rol     esi, 12h
.text:004021B4 xor     eax, esi
.text:004021B6 mov     [edi], eax
.text:004021B8 mov     [edi+10h], ebx
.text:004021BB mov     [edi+20h], ecx
.text:004021BE mov     [edi+30h], edx
```

Maze



```
[+] encrypt data using Salsa20 or ChaCha matches 144
func_addr 0x401e10
          insn_addr: 0x4019a0
          insn_addr: 0x43af40
          insn_addr: 0x401ee2
          insn_addr: 0x401b2e
          insn_addr: 0x43aaf0
          insn_addr: 0x401852
```



```
.text:004372C4                 paddd   xmm6, xmm5
.text:004372C8                 shufps  xmm5, xmm5, 93h ; '“'
.text:004372CC                 shufps  xmm4, xmm3, 24h ; '$'
.text:004372D0                 movaps  xmm3, xmm0
.text:004372D3                 shufps  xmm3, xmm7, 26h ; '&'
.text:004372D7                 pshufd  xmm7, xmm6, 4Eh ; 'N'
.text:004372DC                 pxor    xmm7, xmm4
.text:004372E0                 movdqa  xmm4, xmm7
.text:004372E4                 pslld   xmm7, 10h
.text:004372E9                 psrld   xmm4, 10h
.text:004372EE                 por     xmm7, xmm4
.text:004372F2                 paddd   xmm3, xmm7
.text:004372F6                 pshufd  xmm4, xmm3, 39h ; '9'
.text:004372FB                 xorps   xmm5, xmm4
.text:004372FE                 movaps  xmm4, xmm5
.text:00437301                 pslld   xmm5, 0Ch
.text:00437306                 psrld   xmm4, 14h
.text:0043730B                 por     xmm5, xmm4
.text:0043730F                 pshufd  xmm4, xmm5, 39h ; '9'
.text:00437314                 paddd   xmm4, xmm6
.text:00437318                 pshufd  xmm0, xmm4, 4Eh ; 'N'
.text:0043731D                 pxor    xmm0, xmm7
.text:00437321                 movdqa  xmm6, xmm0
.text:00437325                 pslld   xmm0, 8
.text:0043732A                 psrld   xmm6, 18h
.text:0043732F                 por     xmm0, xmm6
.text:00437333                 paddd   xmm3, xmm0
.text:00437337                 pshufd  xmm6, xmm3, 39h ; '9'
.text:0043733C                 pxor    xmm6, xmm5
.text:00437340                 movdqa  xmm5, xmm6
.text:00437344                 pslld   xmm6, 7
.text:00437349                 psrld   xmm5, 19h
.text:0043734E                 por     xmm6, xmm5
.text:00437352                 pshufd  xmm5, xmm0, 93h ; '“'
.text:00437357                 paddd   xmm4, xmm6
.text:0043735B                 pxor    xmm5, xmm4
.text:0043735F                 movdqa  xmm0, xmm5
.text:00437363                 pslld   xmm5, 10h
.text:00437368                 psrld   xmm0, 10h
.text:0043736D                 por     xmm5, xmm0
.text:00437371                 paddd   xmm3, xmm5
.text:00437375                 pxor    xmm6, xmm3
.text:00437379                 movdqa  xmm0, xmm6
.text:0043737D                 pslld   xmm6, 0Ch
```

Static Malware Analysis in The Next Generation

- Vivisect
  - A combined disassembler/static analysis/symbolic execution/debugger framework
- Capa
  - Detect capabilities in executable files
- Flare-floss
  - Automatically deobfuscate strings from malware binaries

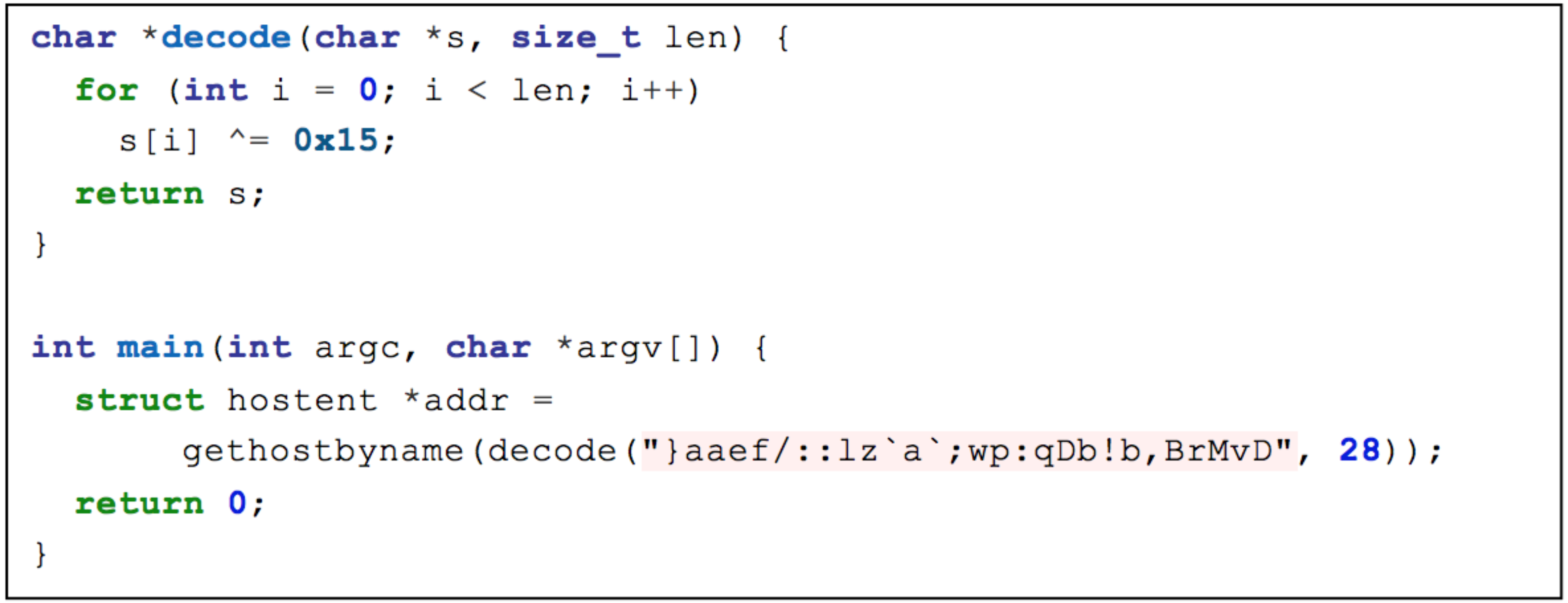
CAPA

“FLOSS”

Flare-floss

Flare-floss

- Beat **strings / grep**
- Solve XOR obfuscation



```
char *decode(char *s, size_t len) {
  for (int i = 0; i < len; i++)
    s[i] ^= 0x15;
  return s;
}

int main(int argc, char *argv[]) {
  struct hostent *addr =
       gethostbyname(decode("}aaef/::lz`a`;wp:qDb!b,BrMvD", 28));
  return 0;
}
```

What to find?

1. static strings (ascii & UTF-16LE)
2. decoded strings
3. stack strings
4. tight strings (in tight loop)

How it works?

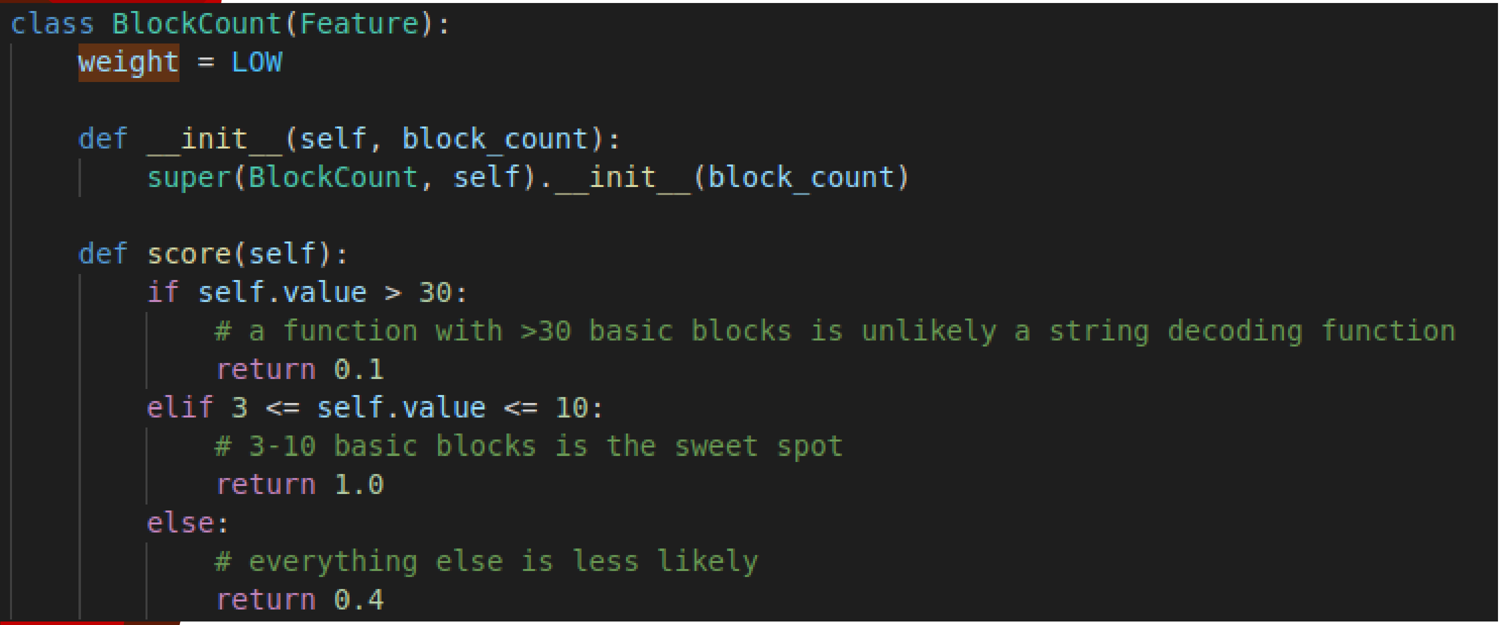
- Based on Vivisect
  - Disassemble and symbolic execution
  - Brute-force emulate all code paths among **basic blocks** and **functions**
    - obtain the arguments passed into a decoding function
- Heuristic scores the likelihood
  - to find potential decoding routines
    - Function contains non-zeroing XOR operation
    - Function has many xrefs
- Snapshot emulator **state** (registers and memory)
  - Emulate decoder functions using emulator state snapshots
  - Compare memory state

Angr

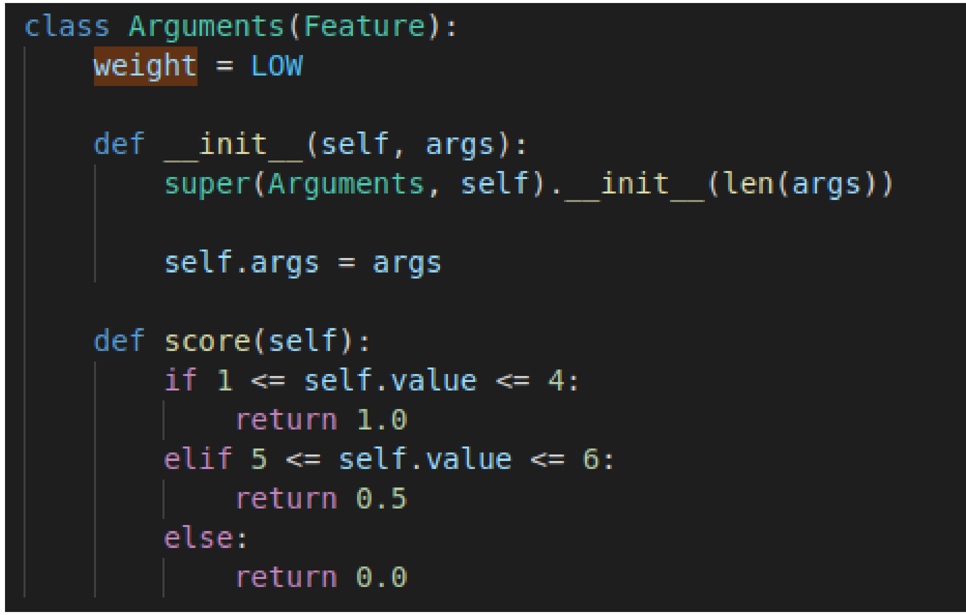
Heuristic Score

LOW = 0.25
MEDIUM = 0.50
HIGH = 0.75
SEVERE = 1.00

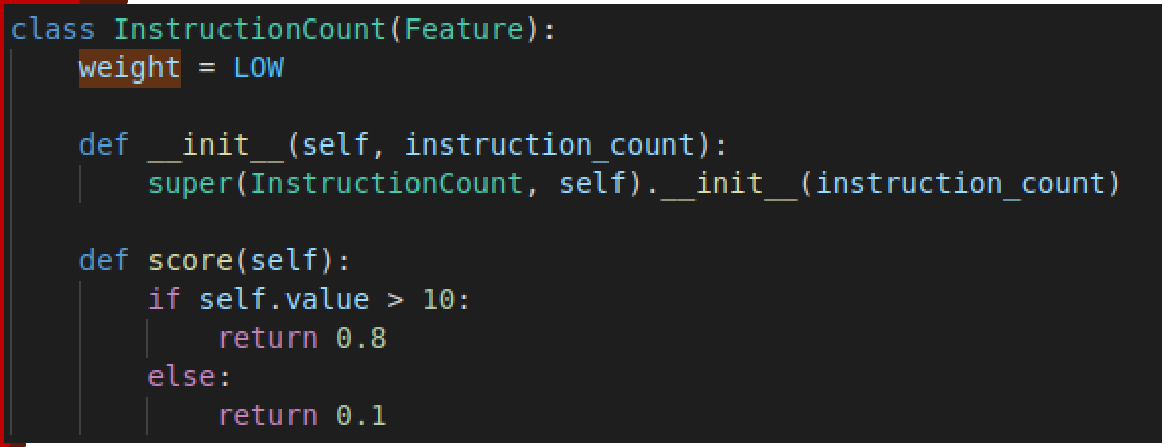
function_features	BlockCount	Low
	InstructionCount	Low
	Arguments	Low
	CallsTo	Medium
	Loop	Medium
	KindaTightLoop	High
basic_block_features	TightLoop	
insn_features	Nzxor	High
	Shift	High
	Mov	Medium
abstract_features	NzxorTightLoop	Severe
	NzxorLoop	Severe



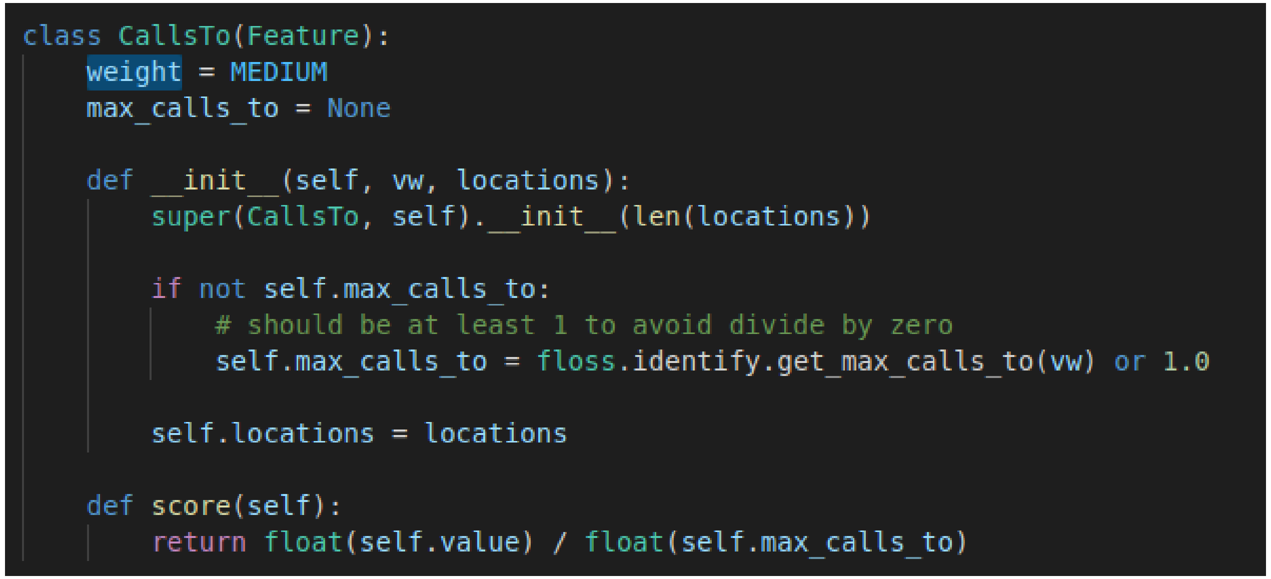
```
class BlockCount(Feature):
    weight = LOW

    def __init__(self, block_count):
        super(BlockCount, self).__init__(block_count)

    def score(self):
        if self.value > 30:
            # a function with >30 basic blocks is unlikely a string decoding function
            return 0.1
        elif 3 <= self.value <= 10:
            # 3-10 basic blocks is the sweet spot
            return 1.0
        else:
            # everything else is less likely
            return 0.4
```



```
class Arguments(Feature):
    weight = LOW

    def __init__(self, args):
        super(Arguments, self).__init__(len(args))

        self.args = args

    def score(self):
        if 1 <= self.value <= 4:
            return 1.0
        elif 5 <= self.value <= 6:
            return 0.5
        else:
            return 0.0
```

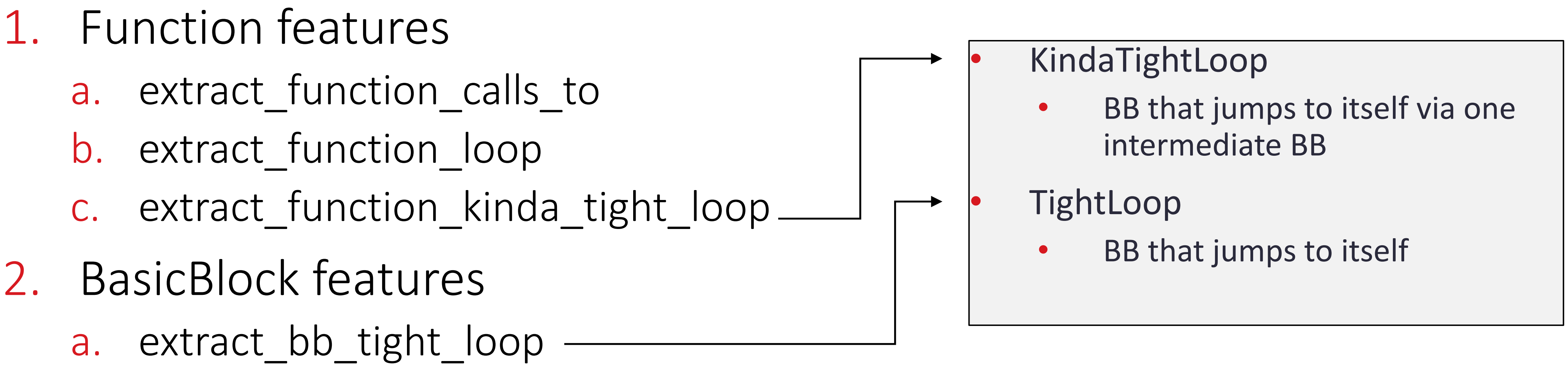


```
class InstructionCount(Feature):
    weight = LOW

    def __init__(self, instruction_count):
        super(InstructionCount, self).__init__(instruction_count)

    def score(self):
        if self.value > 10:
            return 0.8
        else:
            return 0.1
```



```
class CallsTo(Feature):
    weight = MEDIUM
    max_calls_to = None

    def __init__(self, vw, locations):
        super(CallsTo, self).__init__(len(locations))

        if not self.max_calls_to:
            # should be at least 1 to avoid divide by zero
            self.max_calls_to = floss.identify.get_max_calls_to(vw) or 1.0

        self.locations = locations

    def score(self):
        return float(self.value) / float(self.max_calls_to)
```

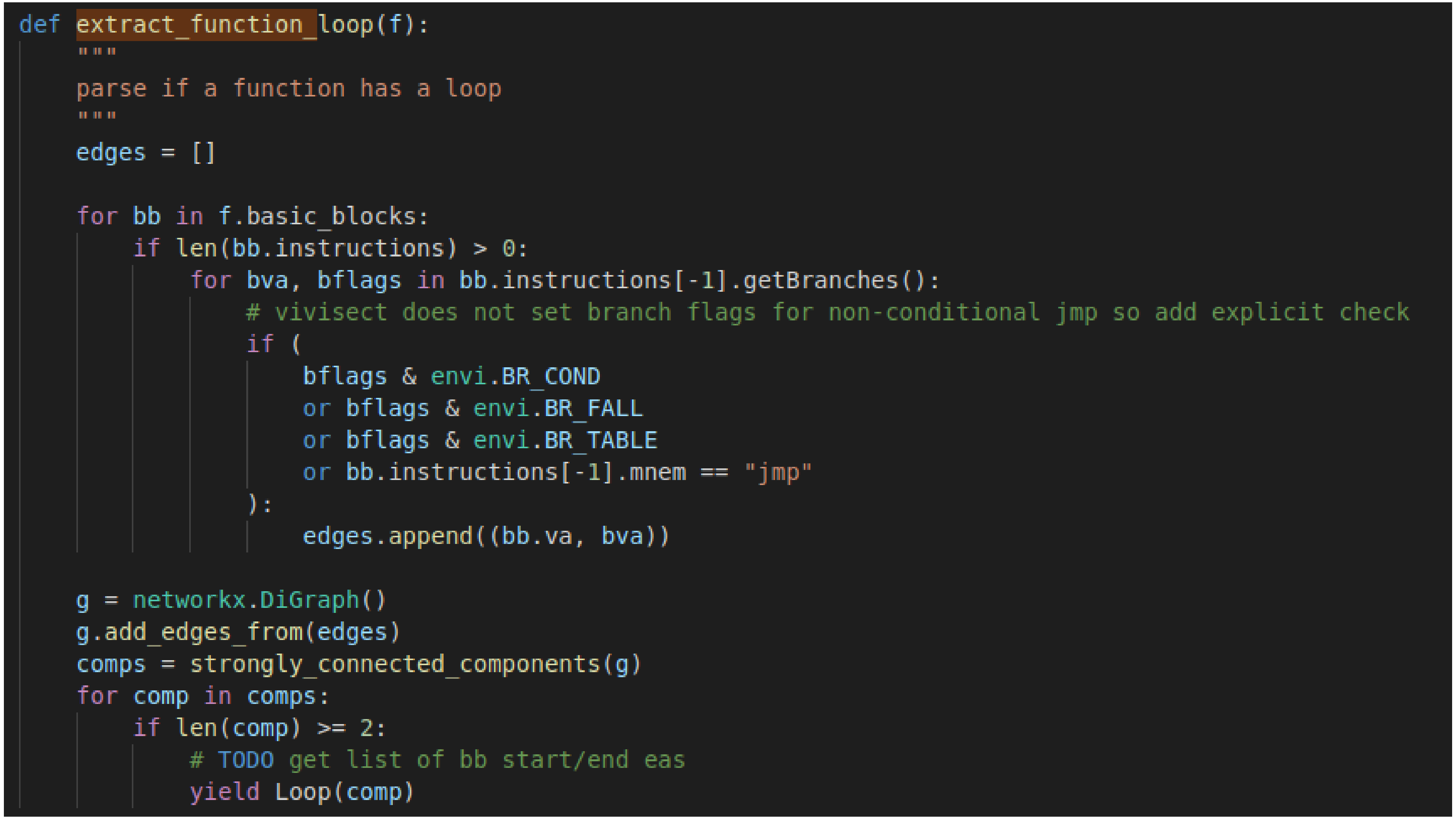
Handler of Extract Features



1. Function features
   a. extract_function_calls_to
   b. extract_function_loop
   c. extract_function_kinda_tight_loop
2. BasicBlock features
   a. extract_bb_tight_loop
3. Insn features
4. Abstract features

- KindaTightLoop
  - BB that jumps to itself via one intermediate BB
- TightLoop
  - BB that jumps to itself

Loop Reconstruct

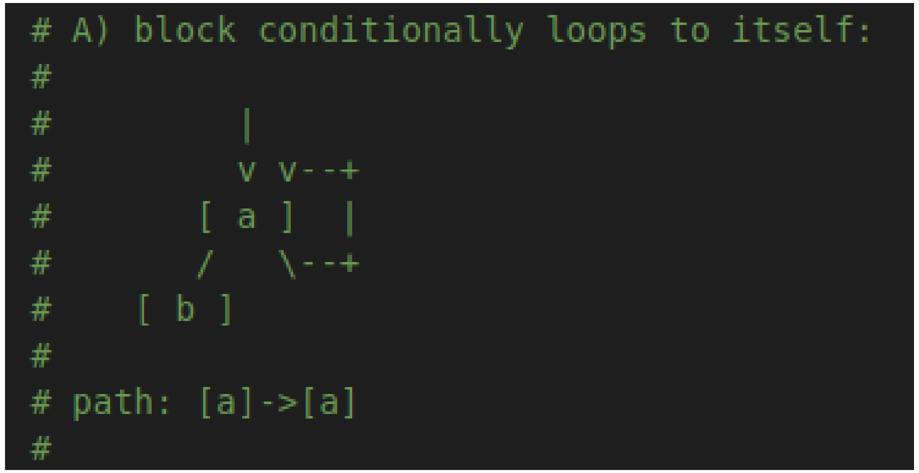


```
def extract_function_loop(f):
    """
    parse if a function has a loop
    """
    edges = []

    for bb in f.basic_blocks:
        if len(bb.instructions) > 0:
            for bva, bflags in bb.instructions[-1].getBranches():
                # vivisect does not set branch flags for non-conditional jmp so add explicit check
                if (
                    bflags & envi.BR_COND
                    or bflags & envi.BR_FALL
                    or bflags & envi.BR_TABLE
                    or bb.instructions[-1].mnem == "jmp"
                ):
                    edges.append((bb.va, bva))

    g = networkx.DiGraph()
    g.add_edges_from(edges)
    comps = strongly_connected_components(g)
    for comp in comps:
        if len(comp) >= 2:
            # TODO get list of bb start/end eas
            yield Loop(comp)
```

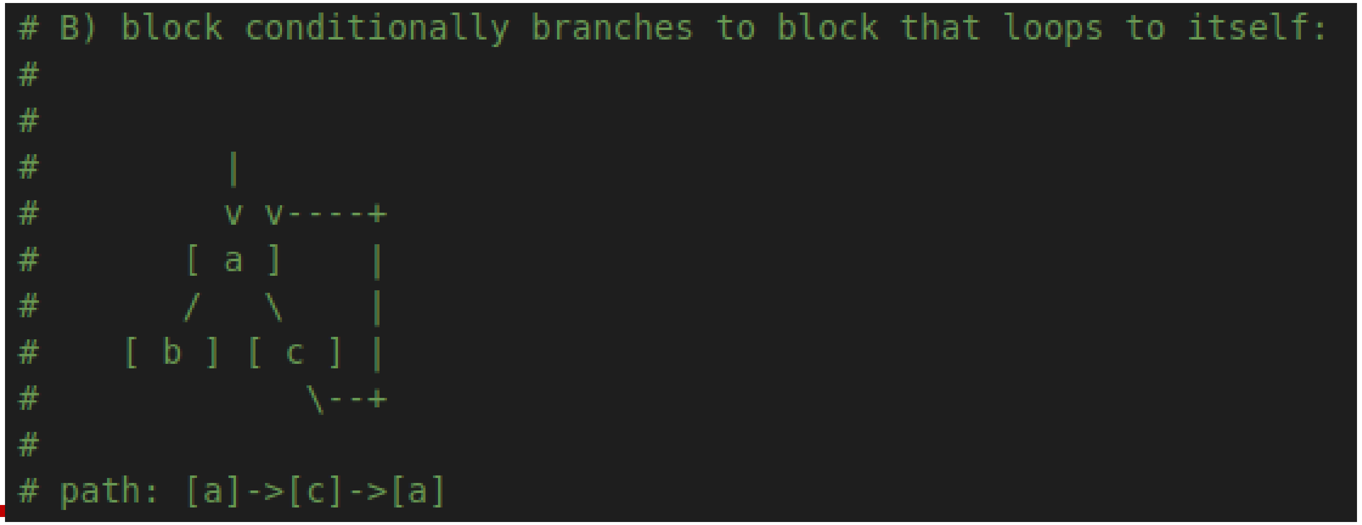
TightLoop Reconstruct

- TightLoop
  - BB that jumps to itself
- KindaTightLoop
  - BB that jumps to itself via one intermediate BB

- Vivisect don’t care the loop
  - but floss care
- skip first and last BBs
- skip blocks that don't have exactly 2 successors
- get the block after loop



```
# A) block conditionally loops to itself:
#
#         |
#         v v--+
#       [ a ]  |
#       /   \--+
#    [ b ]
#
# path: [a]->[a]
#
```

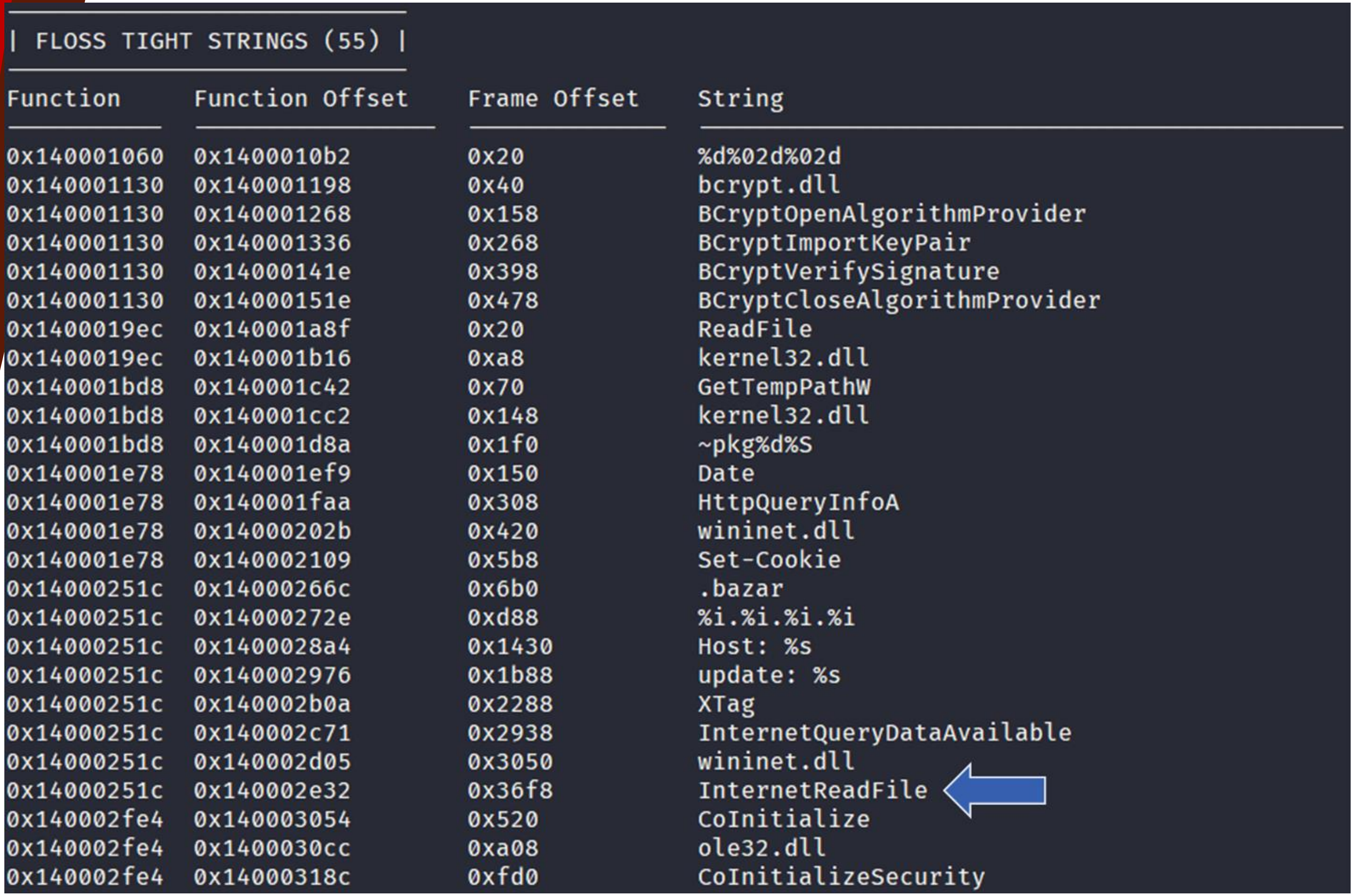


```
# B) block conditionally branches to block that loops to itself:
#
#
#         |
#         v v----+
#       [ a ]    |
#       /   \    |
#    [ b ] [ c ] |
#             \--+
#
# path: [a]->[c]->[a]
```

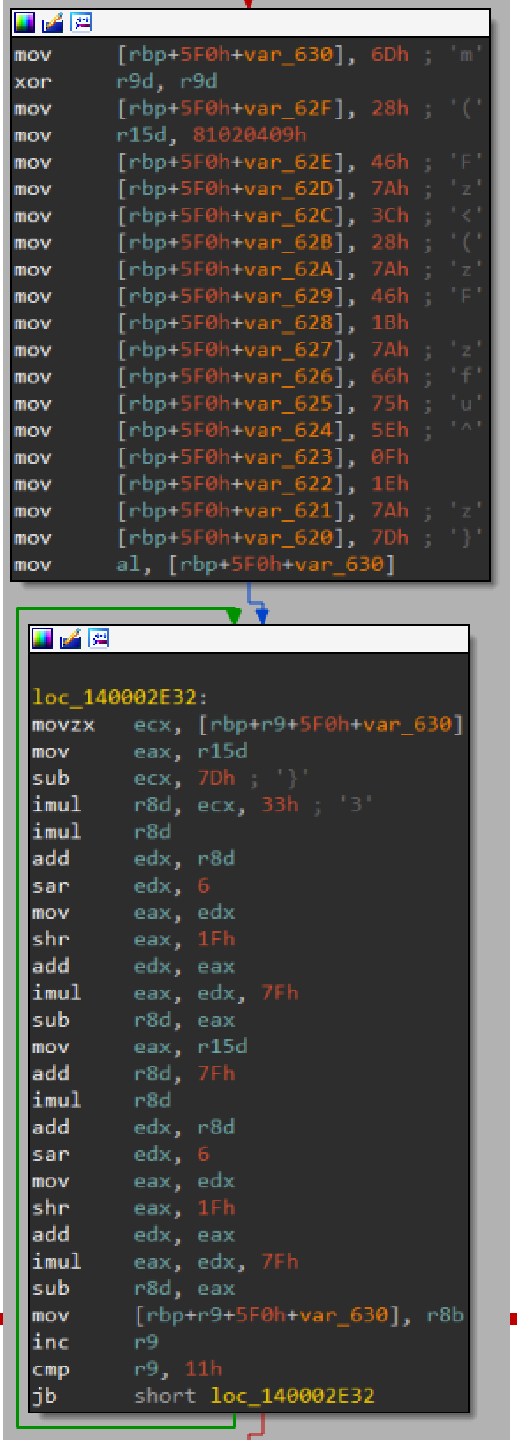
Emulation

1. Brute-force emulate all code paths
2. Find decoding functions
3. Get callers of decoding functions
4. Run the caller while collecting arguments to a decoding function
5. Emulate decoding function and collect snapshots at each interesting place
    - imported API functions
    - the final state of the emulator
6. Extract the delta bytes and turn to strings

Assist Malware Analysis



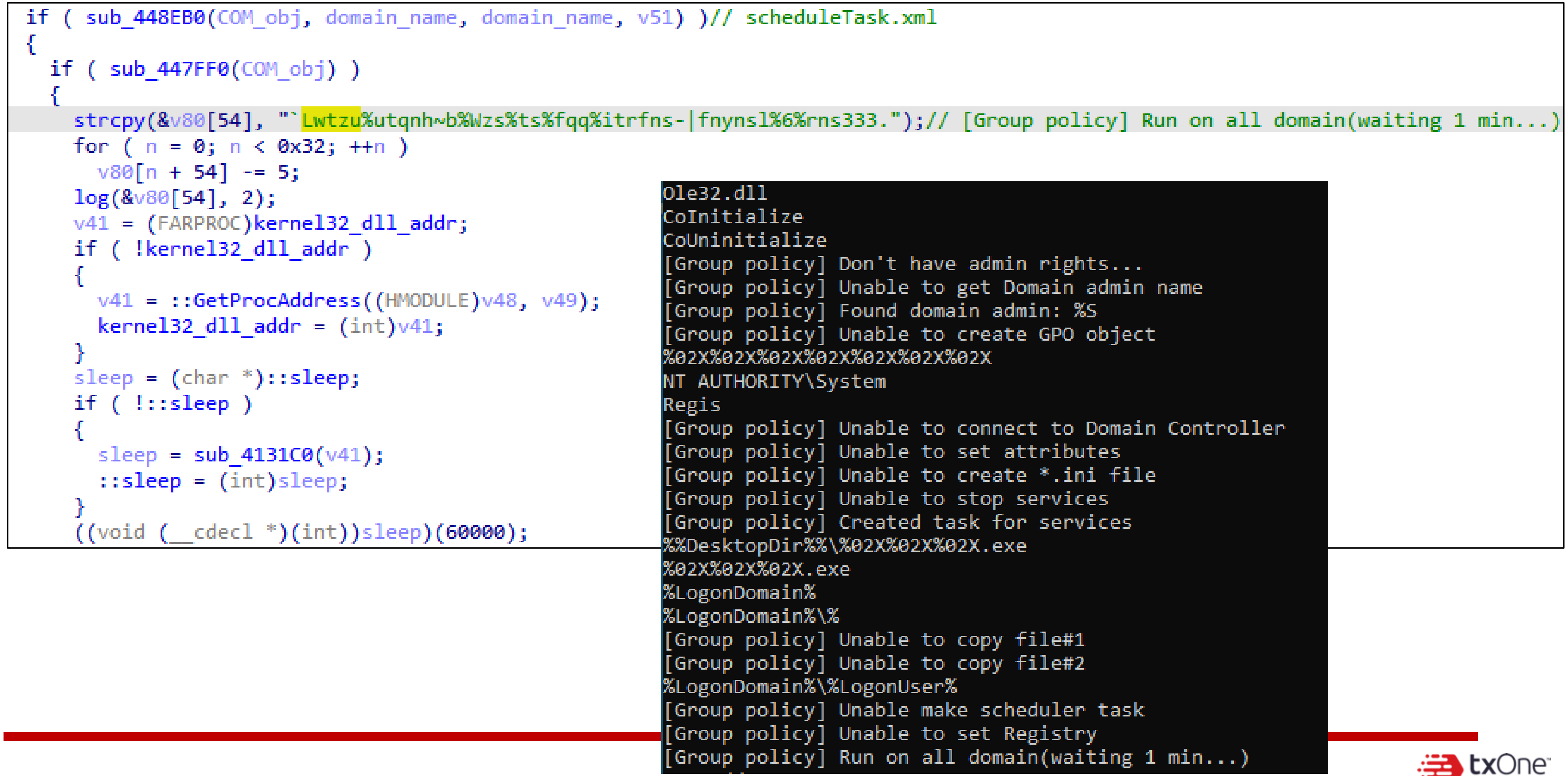
```
 ─────────────────────────
| FLOSS TIGHT STRINGS (55) |
 ─────────────────────────

Function      Function Offset    Frame Offset    String
──────────    ───────────────    ────────────    ─────────────────────────────────────────
0x140001060   0x1400010b2        0x20            %d%02d%02d
0x140001130   0x140001198        0x40            bcrypt.dll
0x140001130   0x140001268        0x158           BCryptOpenAlgorithmProvider
0x140001130   0x140001336        0x268           BCryptImportKeyPair
0x140001130   0x14000141e        0x398           BCryptVerifySignature
0x140001130   0x14000151e        0x478           BCryptCloseAlgorithmProvider
0x1400019ec   0x140001a8f        0x20            ReadFile
0x1400019ec   0x140001b16        0xa8            kernel32.dll
0x140001bd8   0x140001c42        0x70            GetTempPathW
0x140001bd8   0x140001cc2        0x148           kernel32.dll
0x140001bd8   0x140001d8a        0x1f0           ~pkg%d%S
0x140001e78   0x140001ef9        0x150           Date
0x140001e78   0x140001faa        0x308           HttpQueryInfoA
0x140001e78   0x14000202b        0x420           wininet.dll
0x140001e78   0x140002109        0x5b8           Set-Cookie
0x14000251c   0x14000266c        0x6b0           .bazar
0x14000251c   0x14000272e        0xd88           %i.%i.%i.%i
0x14000251c   0x1400028a4        0x1430          Host: %s
0x14000251c   0x140002976        0x1b88          update: %s
0x14000251c   0x140002b0a        0x2288          XTag
0x14000251c   0x140002c71        0x2938          InternetQueryDataAvailable
0x14000251c   0x140002d05        0x3050          wininet.dll
0x14000251c   0x140002e32        0x36f8          InternetReadFile
0x140002fe4   0x140003054        0x520           CoInitialize
0x140002fe4   0x1400030cc        0xa08           ole32.dll
0x140002fe4   0x14000318c        0xfd0           CoInitializeSecurity
```



```
mov     [rbp+5F0h+var_630], 6Dh ; 'm'
xor     r9d, r9d
mov     [rbp+5F0h+var_62F], 28h ; '('
mov     r15d, 81020409h
mov     [rbp+5F0h+var_62E], 46h ; 'F'
mov     [rbp+5F0h+var_62D], 7Ah ; 'z'
mov     [rbp+5F0h+var_62C], 3Ch ; '<'
mov     [rbp+5F0h+var_62B], 28h ; '('
mov     [rbp+5F0h+var_62A], 7Ah ; 'z'
mov     [rbp+5F0h+var_629], 46h ; 'F'
mov     [rbp+5F0h+var_628], 1Bh
mov     [rbp+5F0h+var_627], 7Ah ; 'z'
mov     [rbp+5F0h+var_626], 66h ; 'f'
mov     [rbp+5F0h+var_625], 75h ; 'u'
mov     [rbp+5F0h+var_624], 5Eh ; '^'
mov     [rbp+5F0h+var_623], 0Fh
mov     [rbp+5F0h+var_622], 1Eh
mov     [rbp+5F0h+var_621], 7Ah ; 'z'
mov     [rbp+5F0h+var_620], 7Dh ; '}'
mov     al, [rbp+5F0h+var_630]
```

```
loc_140002E32:
movzx   ecx, [rbp+r9+5F0h+var_630]
mov     eax, r15d
sub     ecx, 7Dh ; '}'
imul    r8d, ecx, 33h ; '3'
imul    r8d
add     edx, r8d
sar     edx, 6
mov     eax, edx
shr     eax, 1Fh
add     edx, eax
imul    eax, edx, 7Fh
sub     r8d, eax
mov     eax, r15d
add     r8d, 7Fh
imul    r8d
add     edx, r8d
sar     edx, 6
mov     eax, edx
shr     eax, 1Fh
add     edx, eax
imul    eax, edx, 7Fh
sub     r8d, eax
mov     [rbp+r9+5F0h+var_630], r8b
inc     r9
cmp     r9, 11h
jb      short loc_140002E32
```

LockBit 2.0



```
if ( sub_448EB0(COM_obj, domain_name, domain_name, v51) )// scheduleTask.xml
{
  if ( sub_447FF0(COM_obj) )
  {
    strcpy(&v80[54], "`Lwtzu%utqnh~b%Wzs%ts%fqq%itrfns-|fnynsl%6%rns333.");// [Group policy] Run on all domain(waiting 1 min...)
    for ( n = 0; n < 0x32; ++n )
      v80[n + 54] -= 5;
    log(&v80[54], 2);
    v41 = (FARPROC)kernel32_dll_addr;
    if ( !kernel32_dll_addr )
    {
      v41 = ::GetProcAddress((HMODULE)v48, v49);
      kernel32_dll_addr = (int)v41;
    }
    sleep = (char *)::sleep;
    if ( !::sleep )
    {
      sleep = sub_4131C0(v41);
      ::sleep = (int)sleep;
    }
    ((void (__cdecl *)(int))sleep)(60000);
```

```
Ole32.dll
CoInitialize
CoUninitialize
[Group policy] Don't have admin rights...
[Group policy] Unable to get Domain admin name
[Group policy] Found domain admin: %S
[Group policy] Unable to create GPO object
%02X%02X%02X%02X%02X%02X%02X
NT AUTHORITY\System
Regis
[Group policy] Unable to connect to Domain Controller
[Group policy] Unable to set attributes
[Group policy] Unable to create *.ini file
[Group policy] Unable to stop services
[Group policy] Created task for services
%%DesktopDir%%\%02X%02X%02X.exe
%02X%02X%02X.exe
%LogonDomain%
%LogonDomain%\%
[Group policy] Unable to copy file#1
[Group policy] Unable to copy file#2
%LogonDomain%\%LogonUser%
[Group policy] Unable make scheduler task
[Group policy] Unable to set Registry
[Group policy] Run on all domain(waiting 1 min...)
```

Conclusion

Sound Bytes

1. 傳統靜態程式分析工具雖然可以快速建立病毒特徵碼，但是未能將資訊充分提取，在混淆跟變種後的識別能力更是幾乎為零
2. Capa能夠提取惡意程式中的行為達到分析語意的效果
3. Flare-floss能夠解決Yara rules或strings無法識別字串混淆跟變種的問題

Sheng-Hao Ma

@aaaddress1

Hank Chen

@hank0438