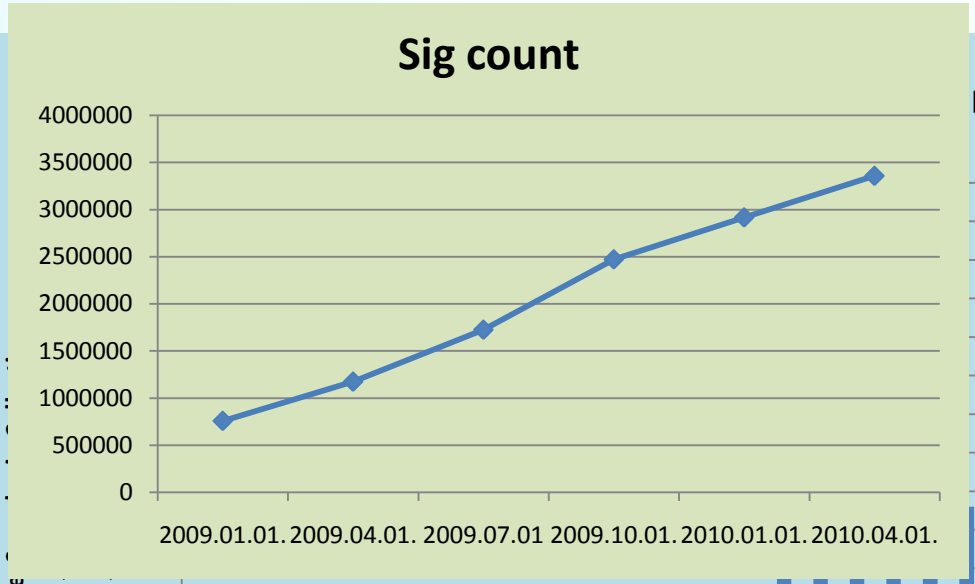
Sequences and Beyond

Gabor Szappanos

VirusBuster Hungary Ltd

Exponentially increasing sample flood



- Signature count, memory usage can not scale with it

Benefits of generic detections

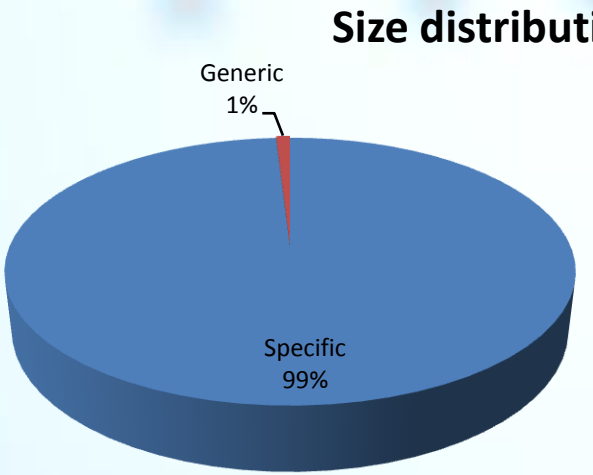
- Help reduce memory footprint
- Family identification
- Less support calls
- … besides, proactive detection

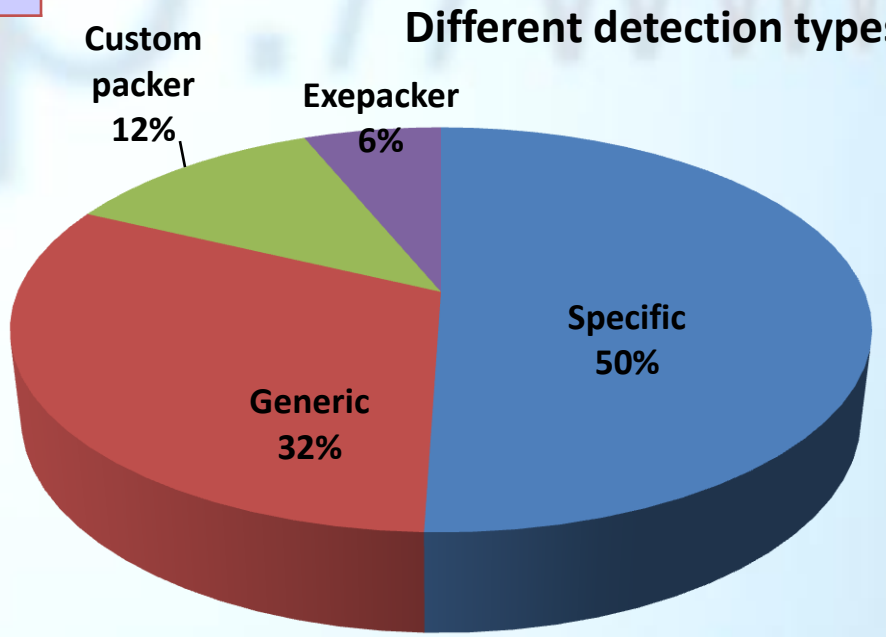
Drawbacks of generic detections

- Suboptimal prevalence stats
- More support calls - difficult generic removal
- Possible false positives
  (<10 so far)

Detection type distribution

Type	Sample detected	Signature count	Efficiency
Specific	9,483,128	1,838,712	5.2
Generic	5,925,781	1,441	4112
Custom packer	2,201,316	400	5503
Exepacker	1,161,114	63	18430

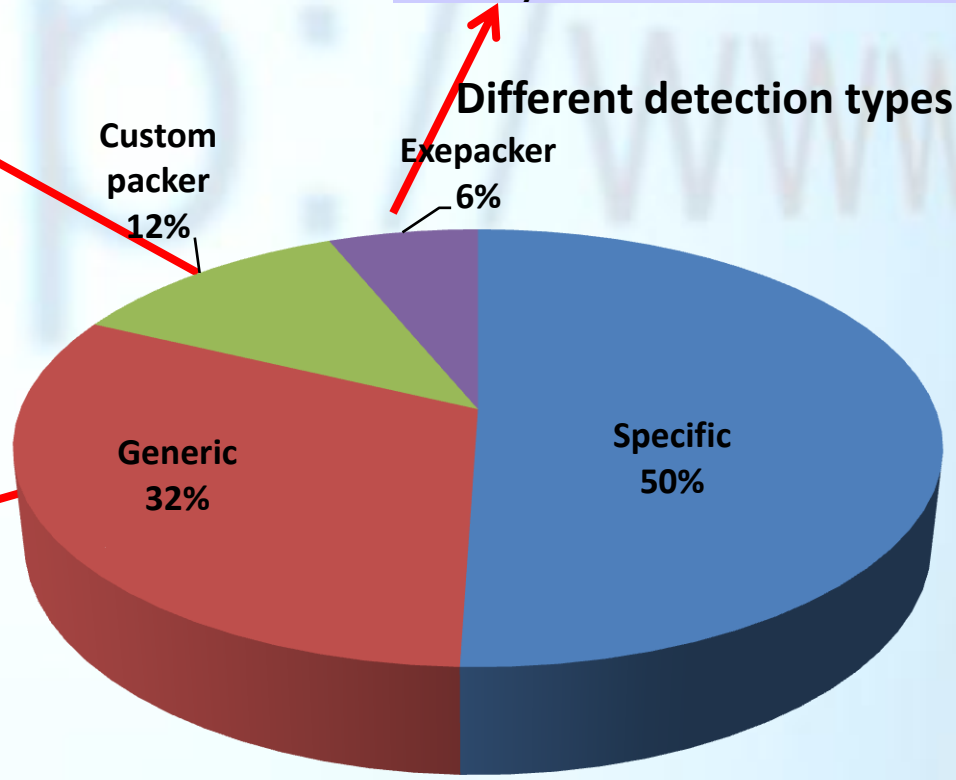




Most prevalent detections

Trojan.DL.Swizzor.Gen!Pac.4	223871
Trojan.DL.Swizzor.Gen!Pac.5	197828
Trojan.Tibs.Gen!Pac.132	96335
Trojan.Lineage.Gen!Pac.3	84864
Adware.Vundo.Gen!Pac.18	83492
Adware.Vundo.Gen!Pac.21	77281
Trojan.Vundo.Gen!Pac.25	68383
Trojan.Vundo.Gen!Pac.31	65159
Trojan.DL.Swizzor.Gen!Pac.3	55766

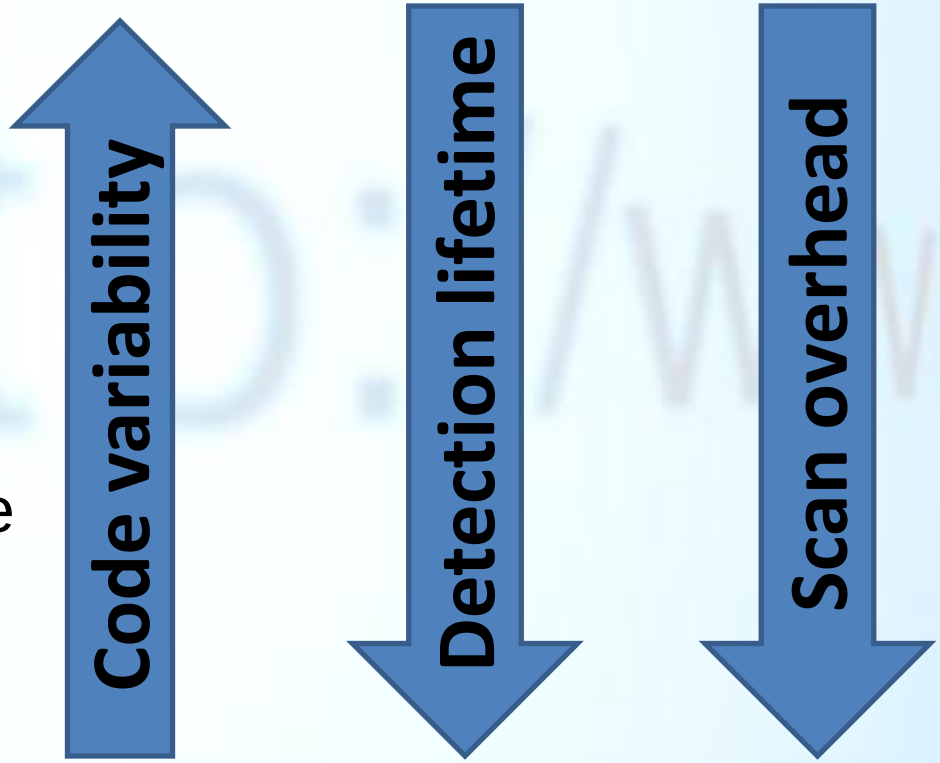
Packed/Upack	460739
Packed/NSPack	165125
Packed/FSG	126669
Packed/UPC	84572
Packed/MEW	32099
Packed/eXPressor	23301
Packed/NSPM	20538
Packed/PolyCrypt	20236
Packed/Themida	20031



Worm.Allaple.Gen	542377
Trojan.OnlineGames.Gen.107	290827
Dialer.Zugang.Gen	223782
Dialer.Agent.Gen	201275
Win32.Virut.Gen	200822
Trojan.OnlineGames.Gen.45	193061
Win32.Virut.Y.Gen	176421
Dialer.eConnect.Gen	167000

Detection source level

- Top level cryptor layer
- Unpacked assembly code
- Malware source code
- Malware API call sequence
- Malware API call tree



Detection approaches

- Pattern matching – sequence/sequence++
- Cryptoanalysis
- API trace
- Algorithmic detection
- Emulation
- Heuristics
- Behaviour analysis

Top detection list

Malware name	Samples	Data collected from	Detection based on
Worm.Allaple.Gen	541,791	emulator	Strong sequence
Packed/Upack	448,532	file	11 sequences around entry point
Trojan.OnlineGames.Gen.107	290,742	file	Trigger sequence and algorithmic check for ID signature
Adware.Trymedia.E	240,324	file	Checksum
Win32.Knat.A	231,598	file	Strong sequence
Dialer.Zugang.Gen	223,781	file	2 Trigger sequences, EVAL and algorithmic check for ID signature
Trojan.DL.Swizzor.Gen!Pac.4	222,795	file	2 weak sequences + strong algorithmic check
PS-MPC_generic	201,437	file or emulator	3 sequences
Dialer.Agent.Gen	201,208	file	Strong sequence
Win32.Virut.Gen	195,966	both file and emulator	weak sequences + strong algorithmic check

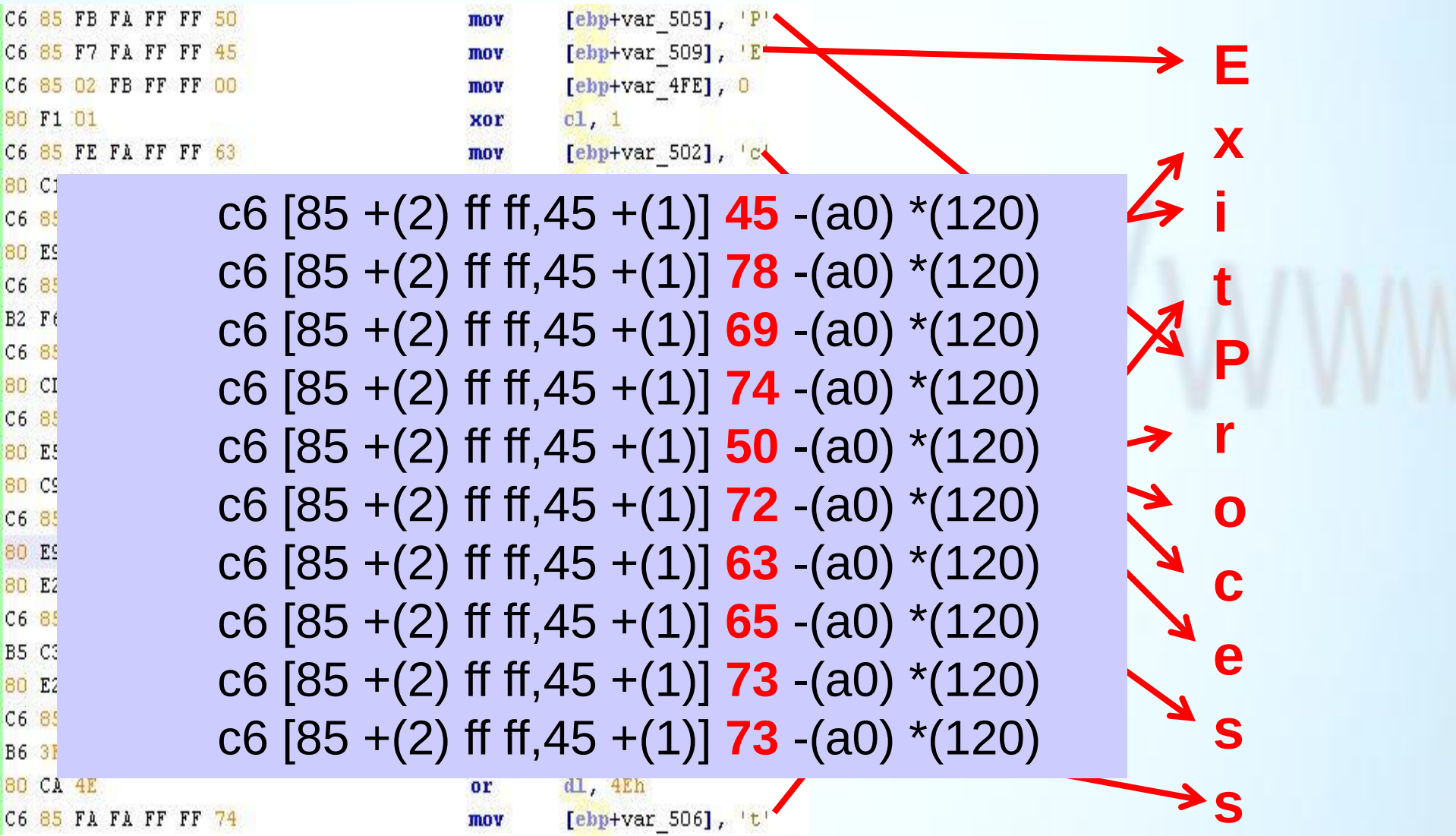
Advantages of pattern matching

- It is always there
- Detection can be releases/updated instantly
- Analysts have experience with it
- Fast – has been optimized for decades
- Surprisingly efficient even nowadays

Basic sequence features

Features / regex tokens	regex notation	VirusBuster notation
literal	literal	6c 69 74 65 72 61 6c
skip one	.	?
skip many (unlimited)	.*	*not supported*
skip many (at most N)	.{,N}	*(N)
forward (exactly N)	.{N}	+(N)
character classes	[0-9a-f]	[30-39 , 61-66]
alternation	(cat | dog)	[63 61 74, 64 67]

Basic sequence features example



Advanced sequence features

Special features	Description	VirusBuster notation
bitmask	(…)	{N … } (N = 0..F)
capturing	(…)	{N … } (N = 0..F)
backreference	\N or $N (N = 1…9)	$N (N = 0..F)
reload	ensure the signature to fit in buffer	++(N) or --(N)
negative sequence	Buffer not matching the sequence	(! pattern)
follow an offset	jump to a previously captured offset	++($N) or -–($N)

Advanced sequence features example 1

```
mov   ecx, 2DC1Fh
push  ecx   ; dwSize
xor   eax, eax
push  eax   ; lpAddress
call  ds:VirtualAlloc

mov   edi, 2DC1Fh ; dwSize
loc_40114E:
and   ch, 0B6h
mov   cl, [esi+edx]
mov   [edx], cl
jnz   short loc_40114E
```

```
B? { 0 +(2) [02,03,04] 00 }
5?
33 C?
5?
FF 15 +(2) 4? 00 ...
[B?,C7 [45 ?, 85 ? FF FF FF]] $0

8A [0?, 1?] [00-3F]
88 [0?, 1?]
...75 [C0-F7]
```

- common regular expression engine features:
  - wildcards (?, *)
  - character classes, bitmasks, ranges, alternatives
  - capturing & backreference

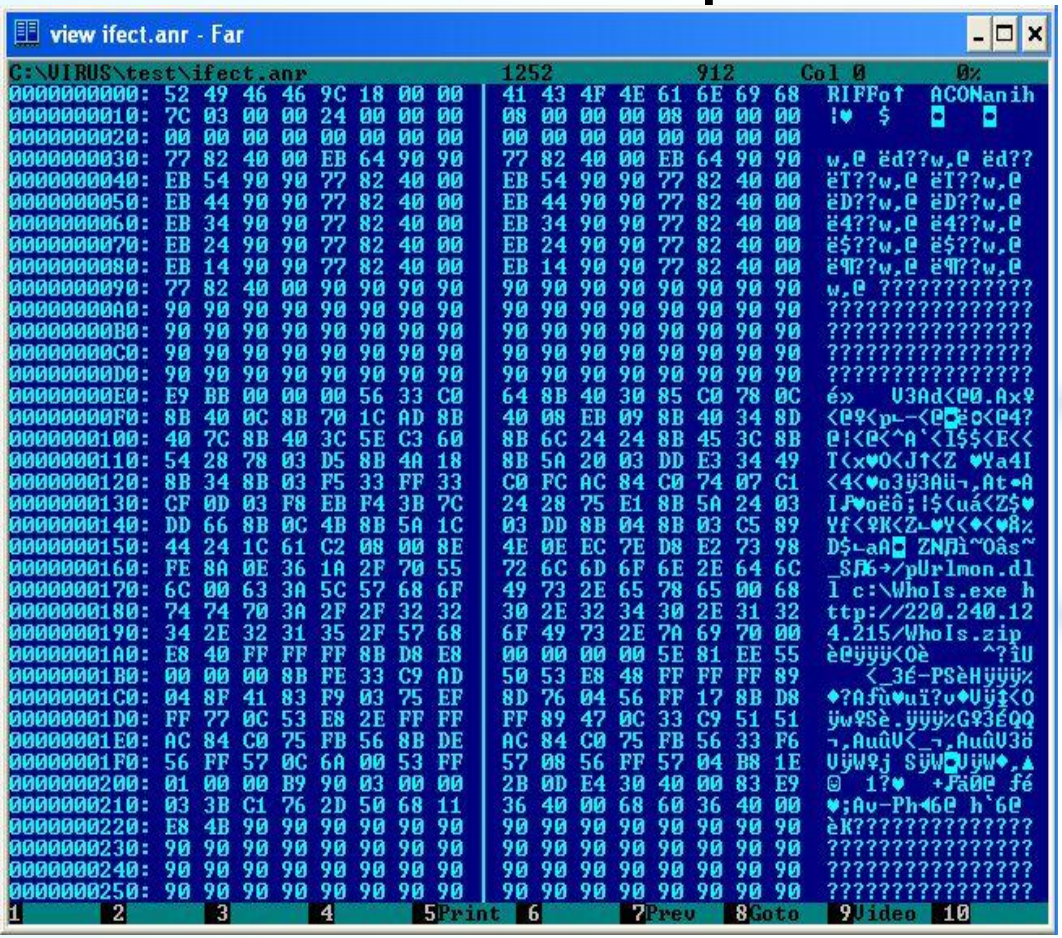
Advanced sequence features example 2

```
83 EC 04          sub     esp, 4
EB 14             jmp     short loc_10001772
...

E9 D7 FD FF FF    jmp     loc_1000154E

...

66 81 F0 86 00    xor     ax, 86h
8A C0             mov     al, al
D2 E0             shl     al, cl
66 03 C1          add     ax, cx
```

```
83 EC 04
EB {0 ?}
+($0)
E9 {1 +(4)}
++($1)
66 81 F0 86 00
8A C0
D2 E0
66 03 C1
```

- Special engine features:
  - Follow an offset

Advanced sequence features example 3

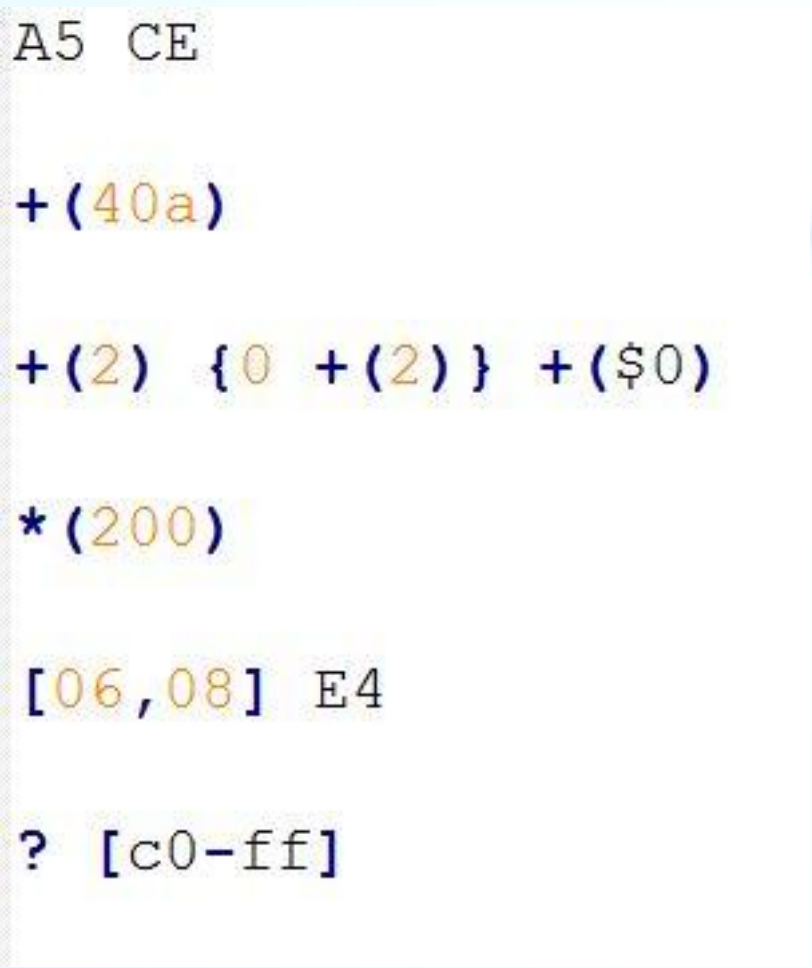



```
52 49 46 46          //RIFF

+(4)

41 43 4F 4E          //ACON

61 6E 69 68          //anih

{!24 00 00 00 }
```

[[[00-23],[25-ff]],
24 [[01-ff], 00
[[01-ff],00 [01-fe]]]]

- Special engine features:
  - Negative sequence

Complex condition – CVE-2006-3647 (format processing)



Magic number

Skip to end of FIB

Skip next structure

Scan the rest

Locate sprmTDefTable

Check size range

All in one – complex condition

```
//PE Executable 80386 architecture
    4d 5a +(3a) {0 +(2)} +($0-3e) 50 45 00 00 4c 01

//with 4 to 6 sections
    {1 [04-06]} 00
+(f0)

//the name of the first section is NOT .text
    {! 2e 74 65 78 74}
+($1*28-4)

//last section atribute is RWX
// 40 00 00 e0
    &?1?????? ? ? &111?????
*(200)

    0f 8? {2 ? [05-0d] 00 00}
++($2)

//Decryptor loop with variable registers
    B? B0 00 00 00                          //mov     eax, 0B0h
    8B ?  1?                                //mov     edi, [eax+edx]
    4?                                      //dec     edi
    6A 0A                                   //push    0Ah

//Reload with large buffer scan
++(4000) *(c000)
//Second decryptor loop
    81 [c0-ff] [00 00 c0 ff, 00 00 40 00]
*(20)
    B? ? [11-17] 00 00 01
*(20)
    81 f? ? 00 00 00
```

Conclusions

- No universal solution
- Have a wide arsenal of possible detection methods
- Generic detection is the hard way
- … but it pays off
- …otherwise you will be flooded by the big numbers
- Look harder for the more efficient detections

Questions?