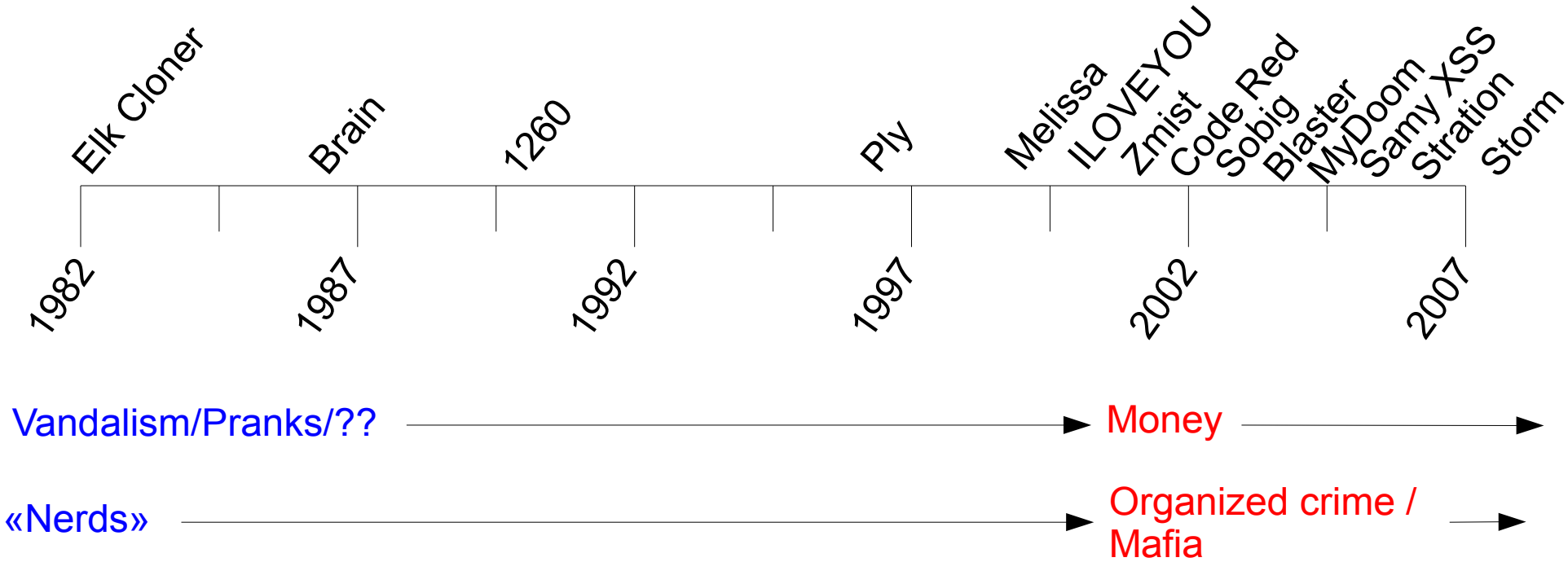
Egil Aspevik Martinsen

Polymorphic Viruses

Material from Master Thesis
«Detection of Junk Instructions in Malicious Software»

History


Elk Cloner
Brain
1260
Ply
Melissa
ILOVEYOU
Zmist
Code Red
Sobig
Blaster
MyDoom
Samy XSS
Stration
Storm
1982
1987
1992
1997
2002
2007
Vandalism/Pranks/??
Money
«Nerds»
Organized crime / Mafia

Two Quotes

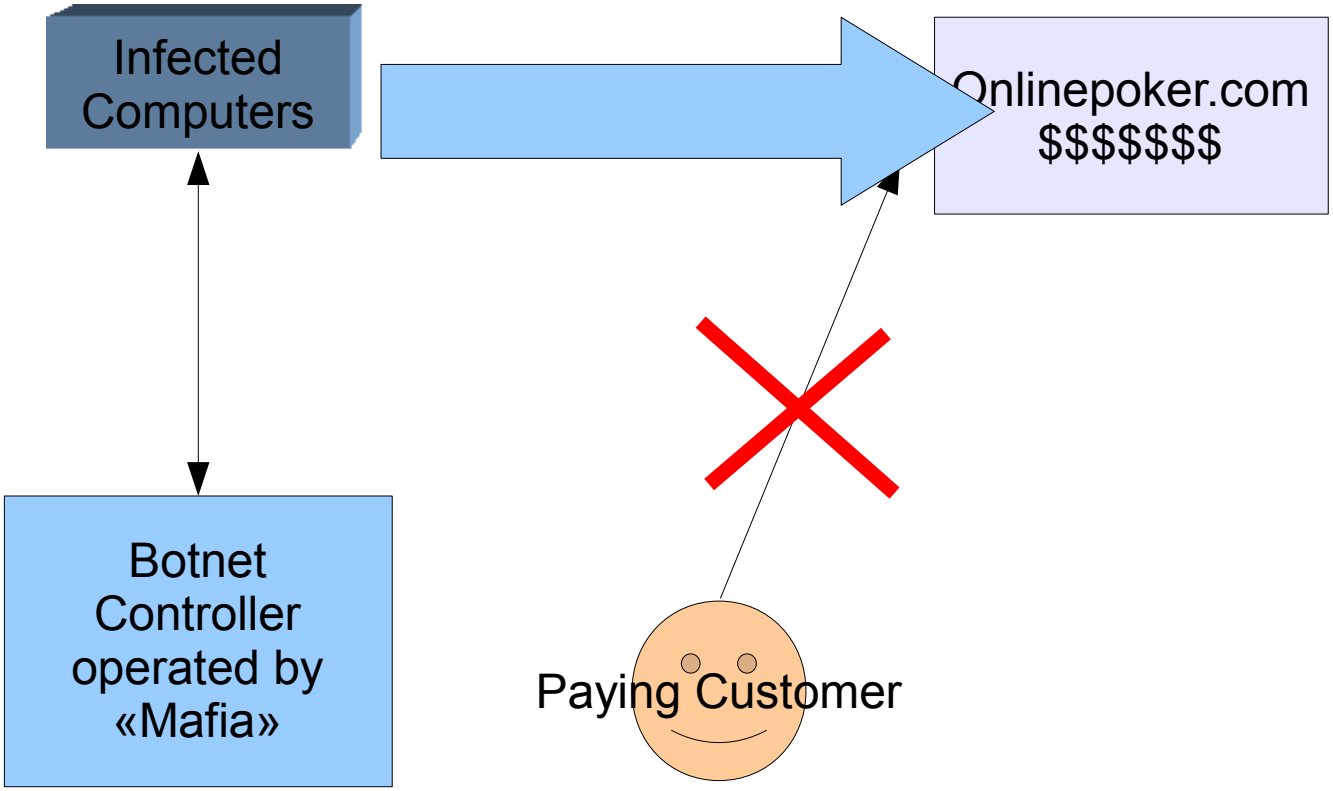
«The most significant change has been the evolution of virus writing hobbyists into criminally operated gangs bent on financial gain»

Mikko Hypponen, F-Secure, 2006,
(http://www.f-secure.com/news/items/news_2006011900.shtml)

«Last year was the first year that proceeds from cybercrime were greater than proceeds from the sales of illegal drugs, and that was, I believe, over $105 billion»

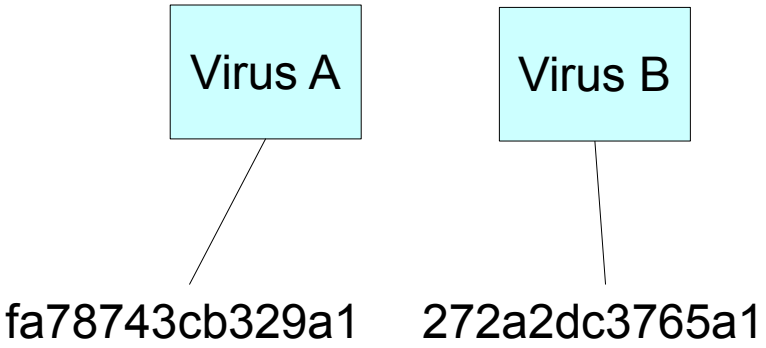
Valerie McNiven, US Treasury, 2006
(«*Dirty money on the wires: the business models of cyber criminals*». Virus Bulletin Conference, 2006)

One «Protection Money» Scheme

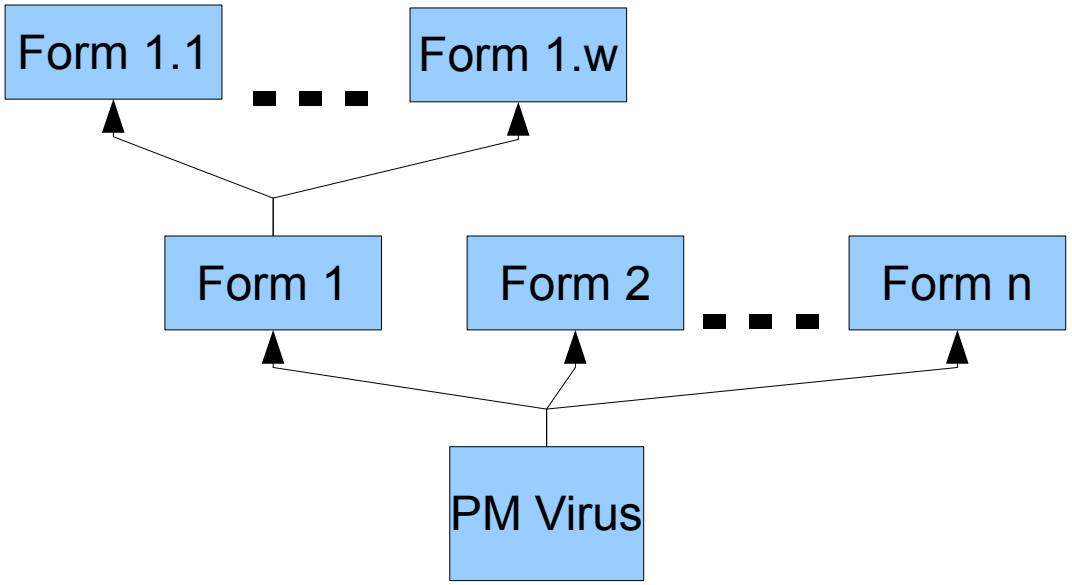


Polymorfism – «many forms»

Regular viruses

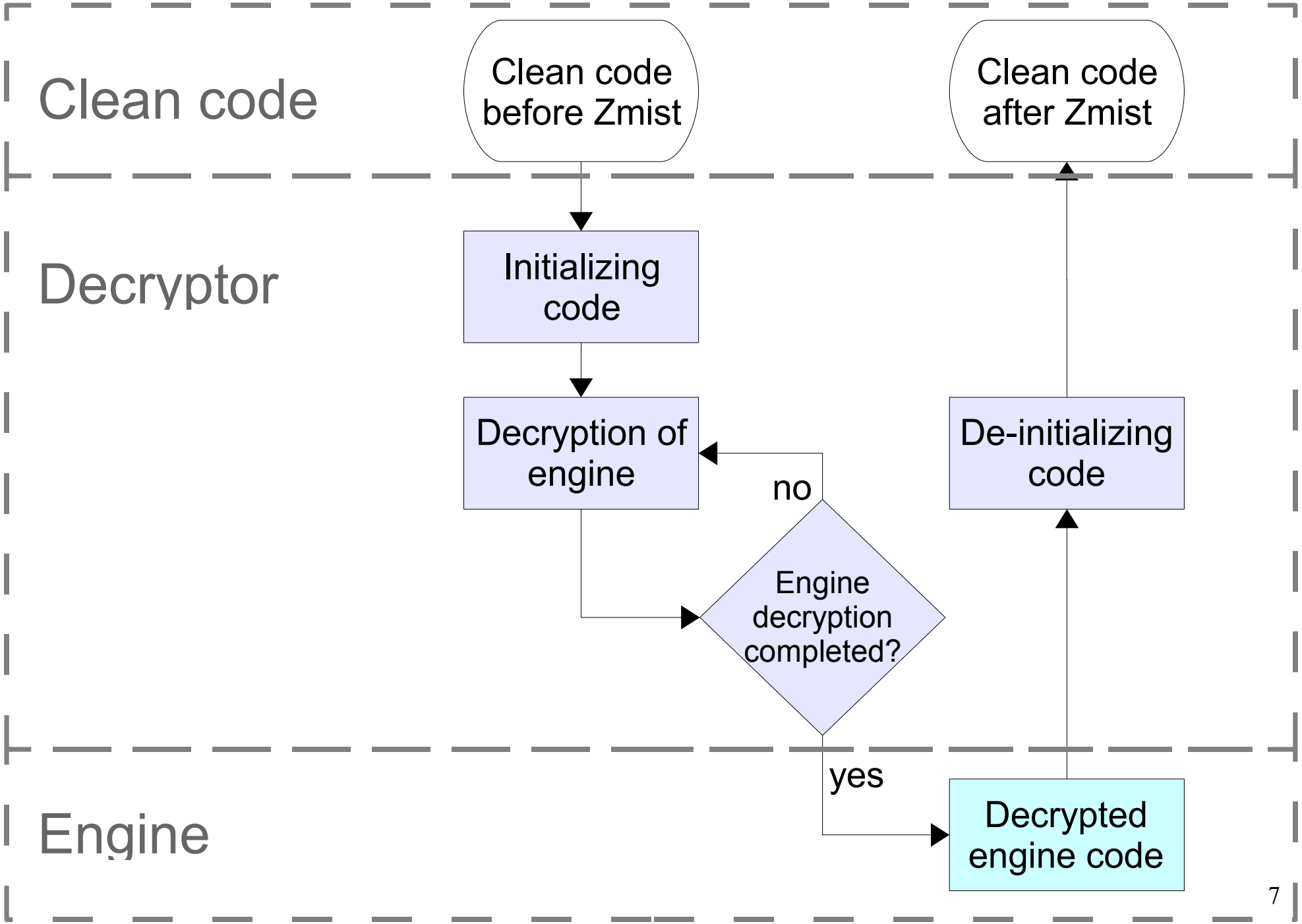


Polymorphic viruses



Case study: Zmist

- Zmist is a virus employing many polymorphic techniques
- Zmist is the only virus known to utilize the code integration technique
- Of the leading 14 antivirus products tested, 7 failed to detect 100% of Zmist samples
  (AV-comparatives.org, February 2008 On-demand comparative)


Clean code
Decryptor
Engine
Clean code before Zmist
Clean code after Zmist
Initializing code
Decryption of engine
Engine decryption completed?
no
yes
De-initializing code
Decrypted engine code

Decryptor 1 (D1)

```
call    [0x4082FE]
mov     [0x41B000], ecx
bsf     ecx, eax
and     ecx, 0x9ACEDD96
mov     [0x41BE98], ebx
mov     ecx, esi
btr     ecx, ecx
mov     ecx, 0x5543994C
mov     ebx, 0x00426758
imul    ecx, edx, 0xA461EDD1
movsx   ecx, ax
mov     [ebx], esi
test    al, 0xBA
test    edi, esi
imul    ecx, esi
mov     esi, 0x00000000
bswap   ebx
mov     [0x424C70], esi
mov     ebx, edi
add     ch, bl
mov     esi, 0x0040A7E1
mov     ecx, [esi]
mov     ebx, ecx
shld    esi, ecx, cl
mov     esi, 0x00423E40
dec     cl
inc     ecx
mov     ecx, esi
push    ebx
```

Decryptor 2 (D2)

```
call    0x00003090
push    eax
pop     [0x415E64]
mov     al, 0x4B
xadd    eax, eax
adc     eax, 0x01D257DA
push    esi
neg     eax
imul    esi, ecx, 0xCD32F179
jmp     0x000030AF
mov     eax, 0x004167FC
not     esi
lea     esi, [-0x5EA21BF8]
mov     esi, esi
bsr     esi, ecx
push    ecx
test    esi, eax
pop     [eax]
mov     esi, 0x60C25CE0
bswap   esi
mov     esi, ebx
movsx   esi, bp
xadd    esi, esi
adc     esi, 0x52ED71CC
jmp     0x000030E0
test    ebx, ecx
mov     ah, 0xF9
mov     esi, 0xFCA091F5
imul    eax, esi, 0x45285F96
```

Decryptor 3 (D3)

```
mov     [0x421634], edi
mov     [0x41FBFC], ebx
mov     edi, 0x00421D90
push    eax
pop     [edi]
mov     [0x41FBF0], 0x00000000
mov     edi, 0x004143E2
push    edi
pop     eax
mov     ebx, [eax]
mov     [0x4211B8], ebx
mov     edi, [0x4035E5]
mov     ebx, edi
mov     [0x41FBF8], ebx
mov     edi, 0x0041FBF0
push    [edi]
pop     eax
mov     ebx, eax
push    ebx
pop     eax
mov     edi, 0x00421630
push    eax
pop     [edi]
mov     eax, 0x00421630
mov     ebx, [eax]
add     ebx, [0x4062A8]
mov     [0x421630], ebx
mov     edi, [0x421630]
mov     ebx, [edi]
```

Polymorphic Techniques

- Arbitrary placement of code and data
- Register renaming
- Change of code flow
- Junk instructions
- Semantically equivalent instructions

Decryptor 1 (D1)

```
call    [0x4082FE]
mov     [0x41B000], ecx
bsf     ecx, eax
and     ecx, 0x9ACEDD96
mov     [0x41BE98], ebx
mov     ecx, esi
btr     ecx, ecx
mov     ecx, 0x5543994C
mov     ebx, 0x00426758
imul    ecx, edx, 0xA461EDD1
movsx   ecx, ax
mov     [ebx], esi
test    al, 0xBA
test    edi, esi
imul    ecx, esi
mov     esi, 0x00000000
bswap   ebx
mov     [0x424C70], esi
mov     ebx, edi
add     ch, bl
mov     esi, 0x0040A7E1
mov     ecx, [esi]
mov     ebx, ecx
shld    esi, ecx, cl
mov     esi, 0x00423E40
dec     cl
inc     ecx
mov     ecx, esi
push    ebx
```

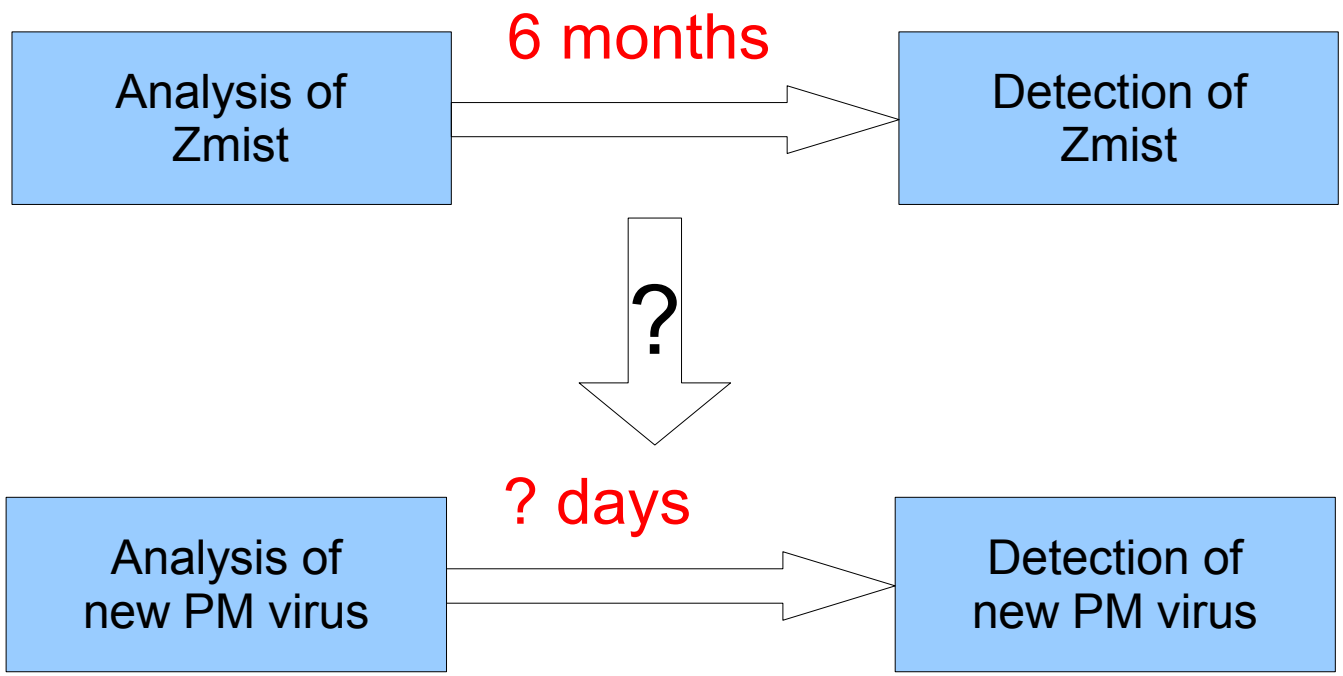
Decryptor 2 (D2)

```
call    0x00003090
push    eax
pop     [0x415E64]
mov     al, 0x4B
xadd    eax, eax
adc     eax, 0x01D257DA
push    esi
neg     eax
imul    esi, ecx, 0xCD32F179
jmp     0x000030AF
mov     eax, 0x004167FC
not     esi
lea     esi, [-0x5EA21BF8]
mov     esi, esi
bsr     esi, ecx
push    ecx
test    esi, eax
pop     [eax]
mov     esi, 0x60C25CE0
bswap   esi
mov     esi, ebx
movsx   esi, bp
xadd    esi, esi
adc     esi, 0x52ED71CC
jmp     0x000030E0
test    ebx, ecx
mov     ah, 0xF9
mov     esi, 0xFCA091F5
imul    eax, esi, 0x45285F96
```

Decryptor 3 (D3)

```
mov     [0x421634], edi
mov     [0x41FBFC], ebx
mov     edi, 0x00421D90
push    eax
pop     [edi]
mov     [0x41FBF0], 0x00000000
mov     edi, 0x004143E2
push    edi
pop     eax
mov     ebx, [eax]
mov     [0x4211B8], ebx
mov     edi, [0x4035E5]
mov     ebx, edi
mov     [0x41FBF8], ebx
mov     edi, 0x0041FBF0
push    [edi]
pop     eax
mov     ebx, eax
push    ebx
pop     eax
mov     edi, 0x00421630
push    eax
pop     [edi]
mov     eax, 0x00421630
mov     ebx, [eax]
add     ebx, [0x4062A8]
mov     [0x421630], ebx
mov     edi, [0x421630]
mov     ebx, [edi]
```

Junk Instruction Detection (JID)



«In fact, while reverse engineering, you can spend up to 80% of your time reading the values in registers and deducing what the code will do or is doing as a result of these values»

Konstantin Rozinov, Bell Labs, (rozinov.sfs.poly.edu/papers/bagle_analysis_v.1.0.pdf)

Decryptor 1 (D1)

```
call    [0x4082FE]
mov     [0x41B000], ecx
bsf     ecx, eax
and     ecx, 0x9ACEDD96
mov     [0x41BE98], ebx
mov     ecx, esi
btr     ecx, ecx
mov     ecx, 0x5543994C
mov     ebx, 0x00426758
imul    ecx, edx, 0xA461EDD1
movsx   ecx, ax
mov     [ebx], esi
test    al, 0xBA
test    edi, esi
imul    ecx, esi
mov     esi, 0x00000000
bswap   ebx
mov     [0x424C70], esi
mov     ebx, edi
add     ch, bl
mov     esi, 0x0040A7E1
mov     ecx, [esi]
mov     ebx, ecx
shld    esi, ecx, cl
mov     esi, 0x00423E40
dec     cl
inc     ecx
mov     ecx, esi
push    ebx
```

Decryptor 2 (D2)

```
call    0x00003090
push    eax
pop     [0x415E64]
mov     al, 0x4B
xadd    eax, eax
adc     eax, 0x01D257DA
push    esi
neg     eax
imul    esi, ecx, 0xCD32F179
jmp     0x000030AF
mov     eax, 0x004167FC
not     esi
lea     esi, [-0x5EA21BF8]
mov     esi, esi
bsr     esi, ecx
push    ecx
test    esi, eax
pop     [eax]
mov     esi, 0x60C25CE0
bswap   esi
mov     esi, ebx
movsx   esi, bp
xadd    esi, esi
adc     esi, 0x52ED71CC
jmp     0x000030E0
test    ebx, ecx
mov     ah, 0xF9
mov     esi, 0xFCA091F5
imul    eax, esi, 0x45285F96
```

Decryptor 3 (D3)

```
mov     [0x421634], edi
mov     [0x41FBFC], ebx
mov     edi, 0x00421D90
push    eax
pop     [edi]
mov     [0x41FBF0], 0x00000000
mov     edi, 0x004143E2
push    edi
pop     eax
mov     ebx, [eax]
mov     [0x4211B8], ebx
mov     edi, [0x4035E5]
mov     ebx, edi
mov     [0x41FBF8], ebx
mov     edi, 0x0041FBF0
push    [edi]
pop     eax
mov     ebx, eax
push    ebx
pop     eax
mov     edi, 0x00421630
push    eax
pop     [edi]
mov     eax, 0x00421630
mov     ebx, [eax]
add     ebx, [0x4062A8]
mov     [0x421630], ebx
mov     edi, [0x421630]
mov     ebx, [edi]
```

Decryptor 1 (D1)

```
call    [0x4082FE]
mov     [0x41B000], ecx
mov     [0x41BE98], ebx
mov     ebx, 0x00426758
mov     [ebx], esi
mov     esi, 0x00000000
mov     [0x424C70], esi
mov     esi, 0x0040A7E1
mov     ecx, [esi]
mov     ebx, ecx
mov     esi, 0x00423E40
mov     ecx, esi
push    ebx
pop     [ecx]
mov     esi, 0x00416723
mov     ebx, [esi]
push    ebx
pop     ecx
push    ecx
pop     [0x424C6C]
mov     esi, 0x00424C70
mov     ebx, [esi]
mov     ecx, ebx
mov     esi, 0x0041B004
push    ecx
pop     [esi]
push    [0x41B004]
pop     ebx
mov     esi, 0x00416559
```

Decryptor 2 (D2)

```
call    0x00003090
push    eax
pop     [0x415E64]
push    esi
jmp     0x000030AF
mov     eax, 0x004167FC
push    ecx
pop     [eax]
jmp     0x000030E0
mov     esi, 0xFCA091F5
sub     esi, 0x2154A1C0
sub     esi, 0xACEEC496
xor     esi, 0x2E5D2B9F
mov     [0x417CF0], esi
push    [0x402A32]
pop     esi
mov     ecx, esi
push    ecx
pop     [0x415E6C]
jmp     0x00003174
push    [0x40A9AB]
jmp     0x00003180
pop     eax
push    eax
pop     [0x415E60]
mov     eax, 0x00417CF0
mov     ecx, [eax]
push    ecx
pop     esi
```

Decryptor 3 (D3)

```
mov     [0x421634], edi
mov     [0x41FBFC], ebx
mov     edi, 0x00421D90
push    eax
pop     [edi]
mov     [0x41FBF0], 0x00000000
mov     edi, 0x004143E2
push    edi
pop     eax
mov     ebx, [eax]
mov     [0x4211B8], ebx
mov     edi, [0x4035E5]
mov     ebx, edi
mov     [0x41FBF8], ebx
mov     edi, 0x0041FBF0
push    [edi]
pop     eax
mov     ebx, eax
push    ebx
pop     eax
mov     edi, 0x00421630
push    eax
pop     [edi]
mov     eax, 0x00421630
mov     ebx, [eax]
add     ebx, [0x4062A8]
mov     [0x421630], ebx
mov     edi, [0x421630]
mov     ebx, [edi]
```

Polymorphism Today

- Serverside polymorphism
  - Pre-mutated
  - Payload only
  - High language polymorphism, not assembler
- Packers
  - Work like Zmist decryptor.
  - Used in clean code to avoid «software cracking»
  - Used in virus code, 3-4 different packers applied