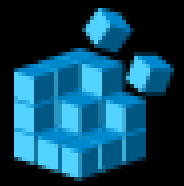
Yet another dev fail ?

Botconf 2021/2022 – Nantes

Hugo RIFFLET – CERT Orange Cyberdefense

@l3m0ntr33

FFF264F9DA03BC03135F472C09218F9D



javax.exe

Uncompress

Nom	Modifié le	Taille
gmobilt.niq	10/07/2019 19:52	117 148 Ko
tbid.scd	10/07/2019 19:52	210 Ko
umicwxg.exe	09/10/2016 17:20	918 Ko

0A33378CA330411EF11D133A25B87DEA

acd.exe

Uncompress

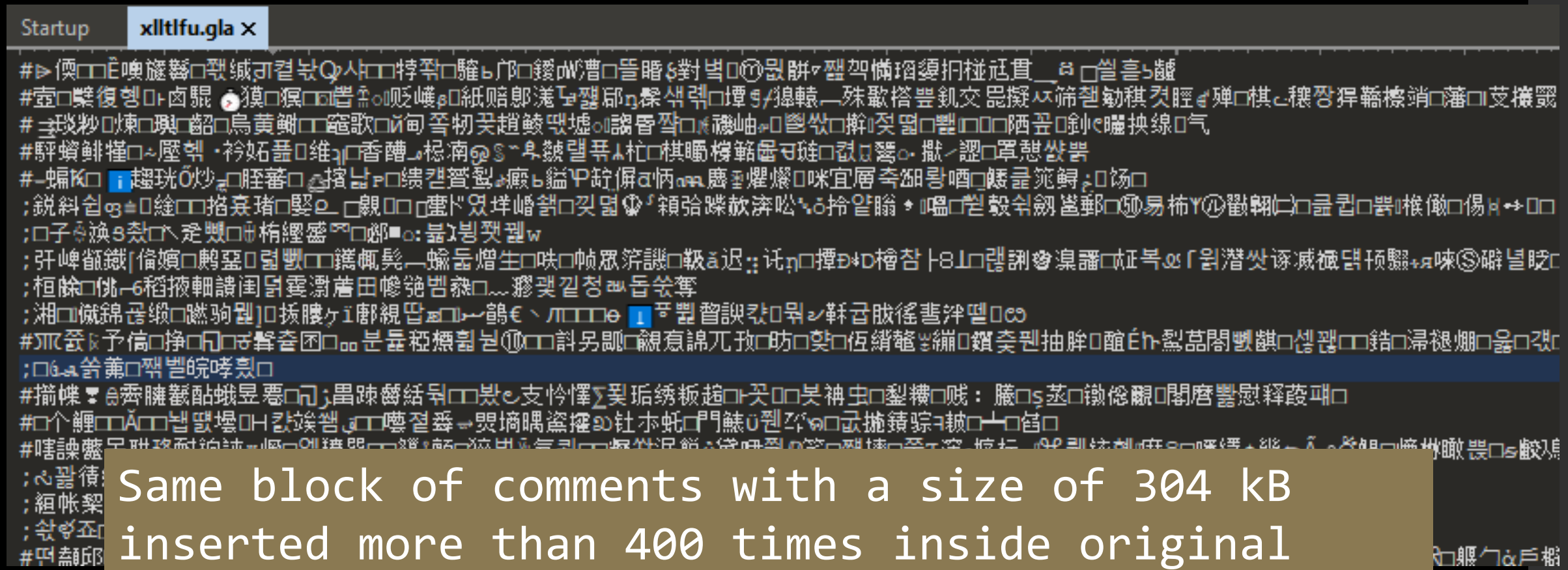
Nom	Modifié le	Taille
facb.rks	06/03/2019 11:46	117 292 Ko
iblkjlkr.tkw	06/03/2019 11:46	210 Ko
ygpimp.exe	09/10/2016 17:20	918 Ko

8815DD6C28C04D90461A9F2E936957FD

lili.exe

Uncompress

Nom	Modifié le	Taille
cipitmke.exe	09/10/2016 16:20	918 Ko
tvvpcqwj.rot	18/05/2019 00:16	115 627 Ko
wfmqpoud.lyp	18/05/2019 00:15	210 Ko


Startup
xlltlfu.gla
Same block of comments with a size of 304 kB inserted more than 400 times inside original script.
Compression algorithm will remove all repeated blocks and thus explain this compression ratio of around 100. Simple but quite effective !

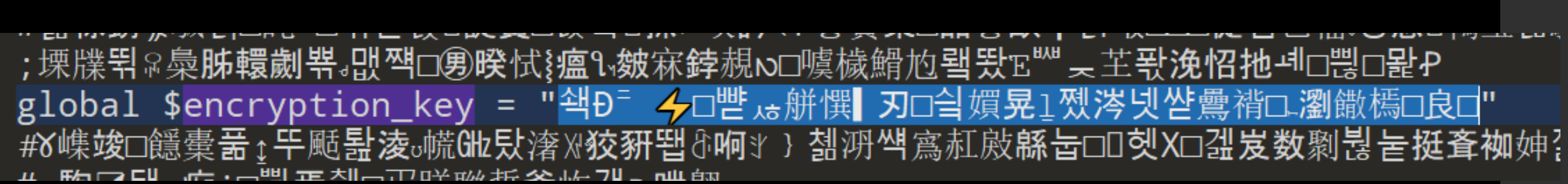
AutoIT script launched by IExpress → `xuzfhxy.exe xlltlfu.gla`

acd.exe

IExpress CAB File

- `xuzfhxy.exe`
  - AutoIT legit interpreter
- `xlltlfu.gla`
  - Obfuscated AutoIt script
- `sayp.jee`
  - Encrypted payload

1. Create persistence folder → `C:\Users\<user>\unpcw\`
2. Move files →
   ```
   ..\xuzfhxy.exe
   ..\xlltlfu.gla
   ..\sayp.jee
   ```
3. Create scheduled task →
   ```
   cmd /c schtasks /create /sc minute  /mo 5 /tn unpcw «xuzfhxy.exe xlltlfu.gla»
   ```
4. Copy .NET RegSvcs.exe to user directory → `../RegSvcs.exe → copied to → C:\Users\<user>\xuzfhxy.exe`
5. Decrypt payload with RC4 → `Decrypted payload is saved in memory`
6. Inject into RegSvcs.exe copy with Process Hollowing technic →
   ```
   API Call flow: CreateProcessW, GetThreadContext, WriteProcessMemory, SetThreadContext, ResumeThread
   ```



Malware dev use code copied from this repository but …
https://github.com/BlizD/AutoIT/blob/master/RC4.au3

```
Func _RC4($Data, $key)
    Local $OPCODE =
    "0xC81001006A006A005356578B551031C989C84989D7F2AE484829C88945F085C00F84DC000000B90001000088C82C0188840DEFFEFFFFE
    F38365F4008365FC00817DFC000100007D478B45FC31D2F775F0920345100FB6008B4DFC0FB68C0DF0FEFFFF01C80345F425FF0000008945
    48B75FC8A8435F0FEFFFF8B7DF486843DF0FEFFFF888435F0FEFFFFFF45FCEBB08D9DF0FEFFFF31FF89FA39550C76638B85ECFEFFFF4025F
    0000008985ECFEFFFF89D80385ECFEFFFF0FB6000385E8FEFFFF25FF0000008985E8FEFFFF89DE03B5ECFEFFFF8A0689DF03BDE8FEFFFF86
    788060FB60E0FB60701C181E1FF0000008A840DF0FEFFFF8B750801D6300642EB985F5E5BC9C21000"
    Local $CodeBuffer = DllStructCreate("byte[" & BinaryLen($OPCODE) & "]")
    DllStructSetData($CodeBuffer, 1, $OPCODE)
    Local $Buffer = DllStructCreate("byte[" & BinaryLen($Data) & "]")
    DllStructSetData($Buffer, 1, $Data)
    VirtualProtect(DllStructGetPtr($CodeBuffer), BinaryLen($OPCODE), 0x40)
    DllCall("user32.dll", "none", "CallWindowProc", "ptr", DllStructGetPtr($CodeBuffer), "ptr", DllStructGetPtr(
    $Buffer), "int", BinaryLen($Data), "str", $key, "int", 0)
    Local $Ret = DllStructGetData($Buffer, 1)
    $Buffer = 0
    $CodeBuffer = 0
    Return $Ret
EndFunc
```

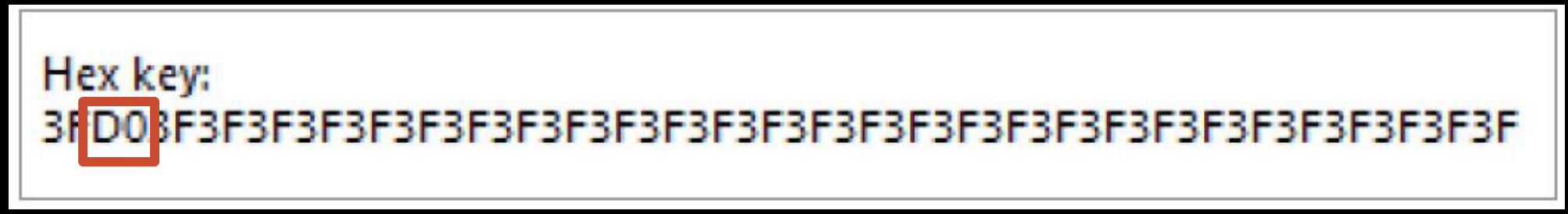
FAIL !!

Use of str instead of wstr

Raw extracted binary key - 30 UTF16 encoded char

```
C5 C1 10 01 93 31 A1 26 3B EA 9F BE F2 D7 6B 44 8F 39 8D 25 03 52 78
0B D8 C2 25 37 43 66 3B 1E B4 CA 94 6D 37 B1 37 C3 0E 4D 99 79 38 E1
87 18 0F 70 4A 99 CA 3B FB E9 7C F9 D5 0A
```

Resulting RC4 encryption key after conversion with “str” - 30 ANSI char



yes… over a 30 bytes key, 29 bytes are just useless !!
NB: AutoIT put 3F which is in fact ‘?’

Thanks !

@l3m0ntr33