Stuxnet et Duqu :
une histoire de codes malveillants mutants

Jean-Yves Marion
LORIA
Université de Lorraine

Jean-Yves.Marion@loria.fr

Loria
Laboratoire lorrain de recherche en informatique et ses applications

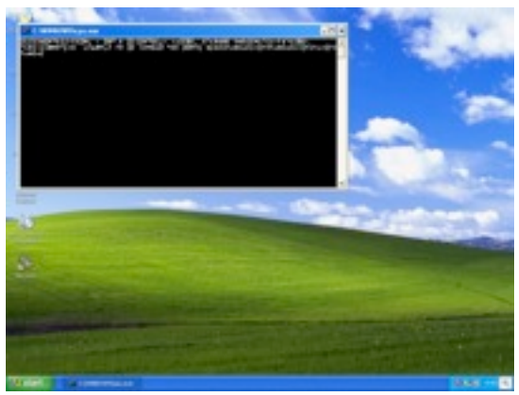
Laboratoire de Haute Sécurité (LHS)

UNIVERSITÉ DE LORRAINE

Stuxnet

Ingénierie Sociale

Infection

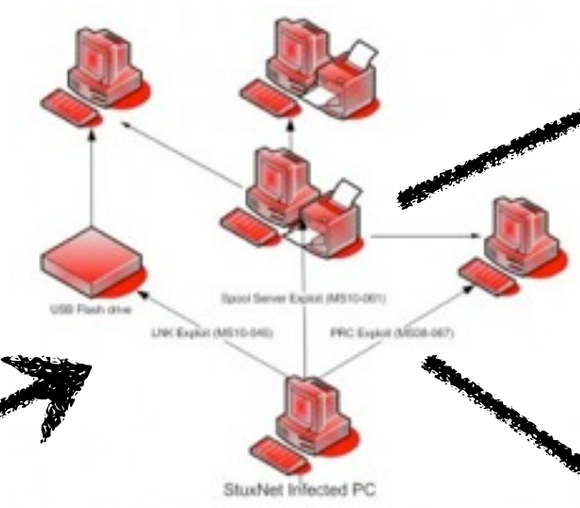


0-day exploit

Faille de sécurité, i.e. un bug

propagation

vol de certificats

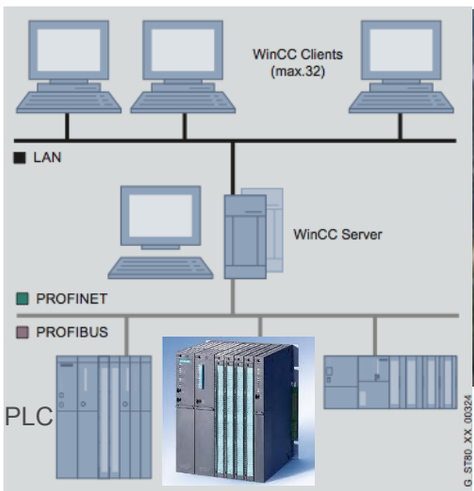


Communication

vol d'informations

mise à jour ...

attaque ciblée



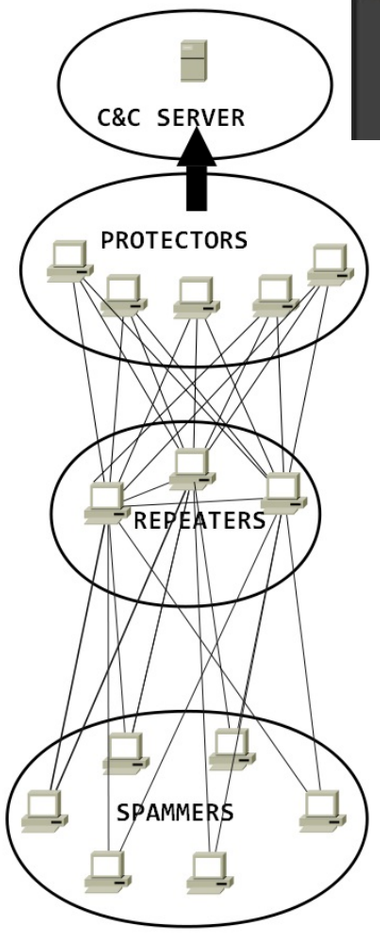
Historique

Début en Juin 2009

Activité Mars et Avril 2010

Connaissance en Juillet 2010

Les attaquants, les cibles, et les défenses



Attaquants : malware

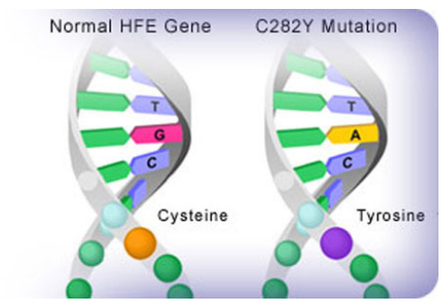
- Botnet
- Virus, vers ...
- Spyware

Cibles :

- Systèmes d'information
  - site web
  - email
  - pdf
  - smartphone
- Systèmes industriels

Défenses : Pare-feux et Anti-virus

Identification (signature) syntaxique

Pro	Cons
Opérationnel 	• Sensible aux mutations/ obfuscations • Inefficace face aux nouvelles attaques

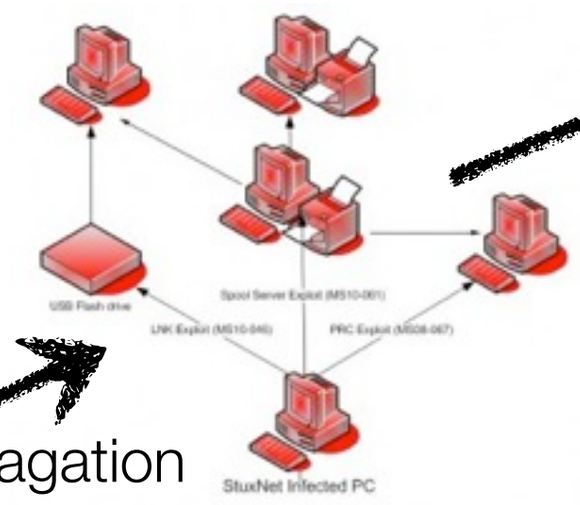
Duqu

Ingénierie Sociale: email

Infection

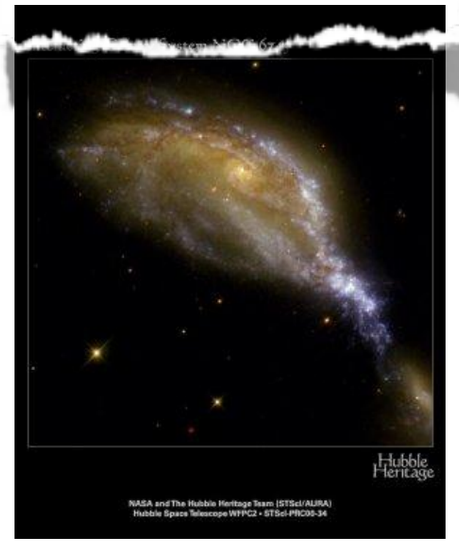
0-day exploit
Faille de sécurité de Word

propagation
vol d'un certificat



attaque ciblée

vol d'informations



Historique
Début Juin 2010 ?
Découverte en Sept. 2011 par Crysys (Budapest)

Rapport de Symantec (confirmé par Kaspersky, Mc Afee, ...) :

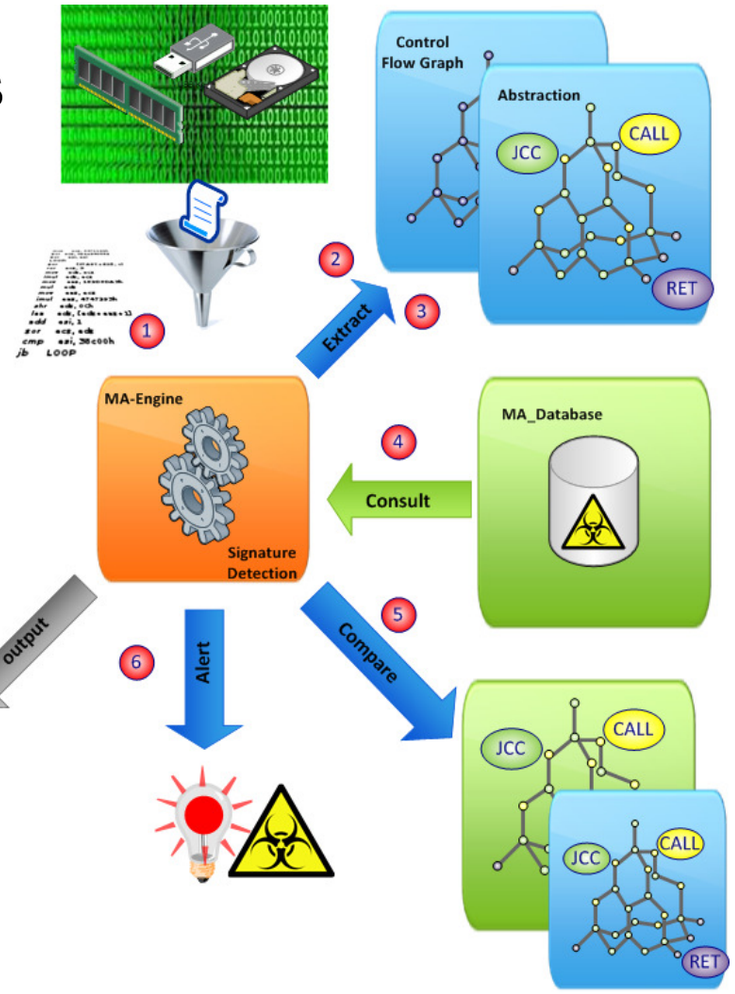
Duqu est similaire à Stuxnet

➡ Même mécanisme d'infection et de propagation
➡ *Mais les anti-virus n'ont rien détecté avant Septembre 2011*
➡ *Aucun des 43 anti-virus de VirusTotal n'a détecté Duqu connaissant Stuxnet.*

Et pourtant

Détection de codes malveillants par analyse morphologique

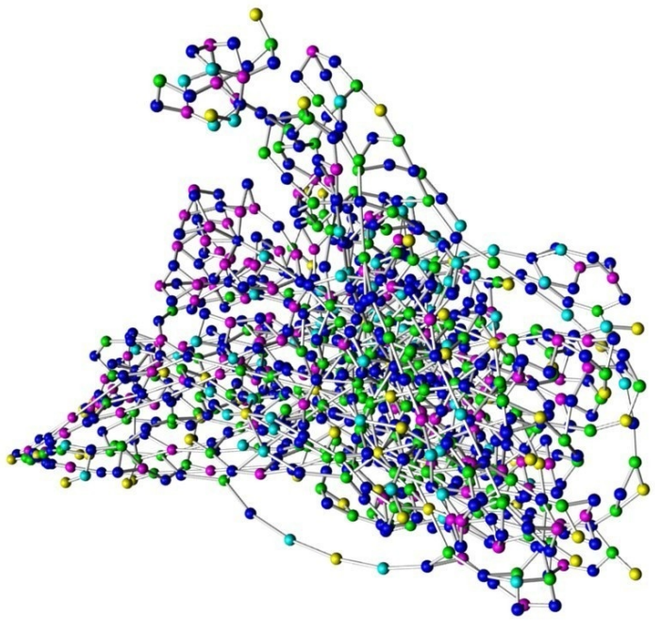
- Anti-virus développé au LHS, Loria
- Architecture partagée avec les anti-virus actuels
- Identification (Signature) différente
- Prototype avancé




```
mov [rbx], eax
push qword [rbp+0x10]
push rax
call 0xddf
mov [rbx+0x4], edi
push qword [rbp+0x18]
pop qword [rbx+0xc]

pop rdi
pop rsi
pop rbx
leave
ret 0x14
mov [rdx], ebx
jmp 0x9
inc dword [rip+0x40812a]
cmp dword [rbx], 0x0
jnz 0x1d
inc dword [rbx]
push qword [rip+0x408146]
call 0x9f5
jmp qword near [rip+0x404070]
pop rbx
leave
ret 0xc
lea eax, [rbp-0x4]
push rax
push 0x0
push rbx
push dword 0x402778
push 0x0
push 0x0
call 0x984
push rax
call 0x95a
mov dword [rip+0x40812a], 0x0
push rsi
call 0xde4
push qword [rbp+0x8]
push dword 0x40814e
call 0x740
push qword [rbp+0x8]
push qword [rbp+0x8]
call 0xfffffffffffff8fb
jmp 0xf
leave
ret 0x4
push dword 0x4057e5
push qword [rbp+0x8]
call 0x66a
push qword [rbp+0xc]
push qword [rbp+0x8]
call 0xfffffffffffff70
jmp 0xa
push qword [rbp-0x4]
call 0x5b3
leave
ret 0x4
push dword 0xafc8
call 0xffffffffffffe106
add eax, 0x1388
invalid
```

```
Sample name: Email-Worm.Win32.Bagl
Number of nodes: 1022
```



Loria Laboratoire lorrain de recherche en informatique et ses applications

Jean-Yves Marion - Laboratoire de Haute Sécurité (LHS)

UNIVERSITÉ DE LORRAINE

Détection de Duqu

Export functions	Numbers of nodes in the MA-signature of `Duqu`	Number of nodes matching `Stuxnet` MA-signature	Ratio	Time in ms
_1	959	605	63.09%	0.06
_2	1458	796	54.60%	0.09
_3	692	476	68.79%	0.04
_4	1212	655	54.04%	0.08
_5	2879	1397	48.52%	0.24
_6	937	613	65.42%	0.07
_7	1998	1079	54.00%	0.14
_8	959	605	63.09%	0.06
Dll EntryPoint	1248	945	75.72%	0.10

Duqu est détecté à partir de Stuxnet par notre anti-virus

➡ *Scan de la mémoire*

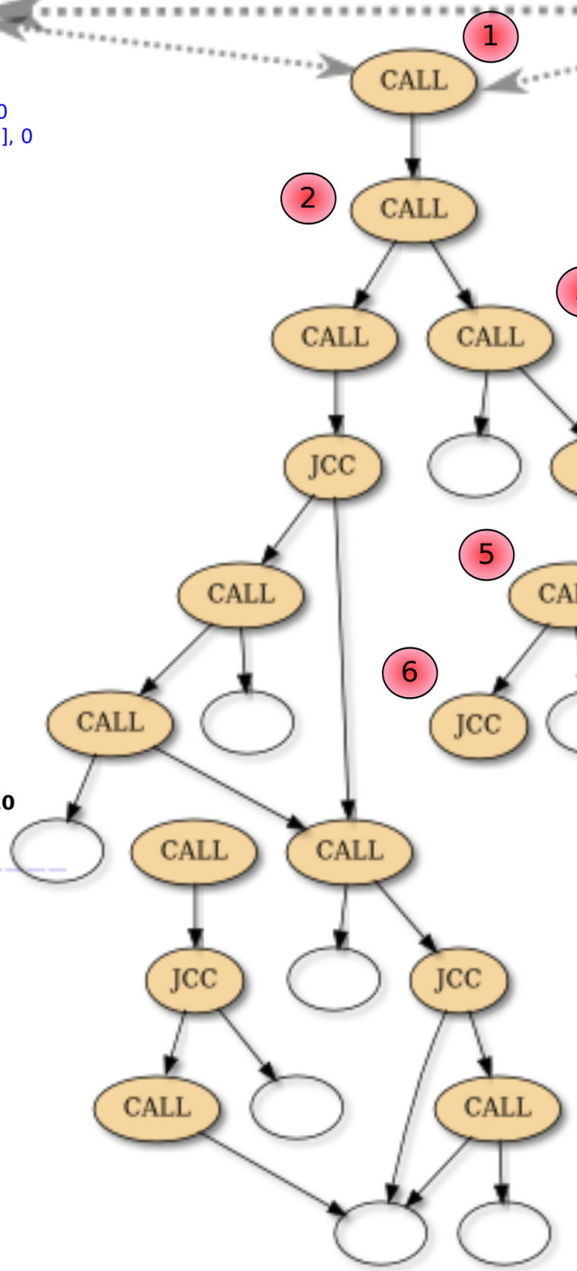
➡ *Détection d'une mutation inconnue (Duqu) à partir de son ancêtre (Stuxnet)*

➡ *Faible taux de faux positifs*

➡ *Reconnaissance des briques logicielles de même fonctionnalité*

Ré-alignement du code

```
.text:100023B4
.text:100023B4
.text:100023B4 sub_100023B4    proc near
.text:100023B4       mov     eax, offset sub_1001ABB6
.text:100023B9       call    sub_10018C14
.text:100023BE       sub     esp, 0Ch
.text:100023C1       push    ebx
.text:100023C2       push    esi
.text:100023C3       and     dword ptr [ebp-4], 0
.text:100023C7       and     dword ptr [ebp-14h], 0
.text:100023CB       lea     eax, [ebp+0Bh]
.text:100023CE       push    eax
.text:100023CF       call    sub_10001318
.text:100023D4       lea     eax, [ebp-10h]
.text:100023D7       push    eax
.text:100023D8       push    dword_1002A134
.text:100023DE       call    sub_10008F32
.text:100023E3       xor     ebx, ebx
.text:100023E5       inc     ebx
.text:100023E6       mov     [ebp-4], ebx
.text:100023E9       push    12Eh
.text:100023EE       lea     eax, [ebp-10h]
.text:100023F1       push    eax
.text:100023F2       lea     eax, [ebp-18h]
.text:100023F5       push    eax
.text:100023F6       call    sub_1000B932
.text:100023FB       add     esp, 0Ch
.text:100023FE       mov     byte ptr [ebp-4], 2
.text:10002402       mov     eax, [eax]
.text:10002404       push    dword ptr [ebp+8]
.text:10002407       mov     ecx, eax
.text:10002409       call    sub_1000BC22
.text:1000240E       mov     [ebp-14h], ebx
.text:10002411       mov     [ebp-4], bl
.text:10002414       mov     esi, [ebp-18h]
.text:10002417       test    esi, esi
.text:10002419       jz      short loc_10002420
.text:1000241B       call    sub_1000239E
.text:10002420
```




```
.text:100048DC       pop     ebx
.text:100048DD       mov     large fs:0, ecx
.text:100048E4       leave
.text:100048E5       retn    4
.text:100048E5 sub_10004890    endp
.text:100048E5
.text:100048E8
.text:100048E8 sub_100048E8    proc near
.text:100048E8
.text:100048E8       mov     eax, offset loc_1004F019
.text:100048ED       call    sub_1004AB94
.text:100048F2       sub     esp, 0Ch
.text:100048F5       push    ebx
.text:100048F6       push    esi
.text:100048F7       push    edi
.text:100048F8       mov     edi, [ebp+8]
.text:100048FB       and     dword ptr [ebp-4], 0
.text:100048FF       and     dword ptr [ebp-14h], 0
.text:10004903       lea     eax, [ebp+13h]
.text:10004906       push    eax
.text:10004907       call    sub_10001E2D
.text:1000490C       lea     eax, [ebp-10h]
.text:1000490F       push    eax
.text:10004910       push    dword_1006A840
.text:10004916       call    sub_100223A7
.text:1000491B       xor     ebx, ebx
.text:1000491D       inc     ebx
.text:1000491E       mov     [ebp-4], ebx
.text:10004921       push    dword ptr [ebp+0Ch]
.text:10004924       lea     eax, [ebp-10h]
.text:10004927       push    eax
.text:10004928       lea     eax, [ebp-18h]
.text:1000492B       push    eax
.text:1000492C       call    sub_10026254
.text:10004931       add     esp, 0Ch
.text:10004934       mov     byte ptr [ebp-4], 2
.text:10004938       mov     eax, [eax]
.text:1000493A       push    ebx
.text:1000493B       push    ebx
.text:1000493C       push    7FFFFFFFh
.text:10004941       push    0FFFFFFFFh
.text:10004943       push    edi
.text:10004944       mov     ecx, eax
.text:10004946       call    sub_10021842
.text:1000494B       mov     [ebp-14h], ebx
.text:1000494E       mov     [ebp-4], bl
.text:10004951       mov     esi, [ebp-18h]
.text:10004954       test    esi, esi
.text:10004956       jz      short loc_1000495D
.text:10004958       call    sub_10001089
.text:1000495D
```

```
.text:10002420 loc_10002420:
.text:10002420       mov     byte ptr [ebp-4], 0
.text:10002424       mov     esi, [ebp-10h]
.text:10002427       test    esi, esi
.text:10002429       jz      short loc_10002430
.text:1000242B       call    sub_1000239E
.text:10002430
.text:10002430 loc_10002430:
.text:10002430       mov     ecx, [ebp-0Ch]
.text:10002433       mov     eax, [ebp+8]
.text:10002436       pop     esi
.text:10002437       pop     ebx
.text:10002438       mov     large fs:0, ecx
.text:1000243F       leave
.text:10002440       retn
.text:10002440 sub_10002 3B4   endp
```

```
.text:1000495D loc_1000495D:
.text:1000495D       cmp     byte ptr [ebp+10h], 0
.text:10004961       jz      short loc_1000497F
.text:10004963       mov     ecx, [edi+4]
.text:10004966       mov     eax, [edi+4]
.text:10004969       mov     eax, [eax+8]
.text:1000496C       xor     edx, edx
.text:1000496E       cmp     [ecx+0Ch], edx
.text:10004971       jle     short loc_1000497F
```

Conclusion

- Les anti-virus connaissaient Stuxnet, et pourtant, ils n'ont pas détecté Duqu.
- Nous savons détecter Duqu par Analyse Morphologique à partir de Stuxnet

- Nécessité d'une recherche indépendante
- Analyse de programmes binaires est difficile:
  - Nécessité d'une recherche fondamentale
  - Besoin d'un cadre pour expérimenter : Laboratoire de haute Sécurité
  - Prototype à rendre opérationnel