

# Techniques avancées de reverse

5d54626cc43ba7ce625903ffa39c049c

Groupe de sécurité de l'information, École normale supérieure

16 octobre 2019



# Rappels de la première fois

- Introduction à la rétro-ingénierie
  - ▶ Définition et cadre juridique
  - ▶ Classification des techniques
  - ▶ Quelques outils fondamentaux (`objdump`, `strings`, `readelf`)
- Introduction à la compilation et à l'assembleur
  - ▶ Structure d'un compilateur (analyse, synthèse)
  - ▶ Fondamentaux de l'assembleur, encodage binaire d'instructions
  - ▶ Abstractions assembleur (ABI, fonctions et conventions d'appel, ...)
  - ▶ Compilation de programmes structurés (boucles, conditions)
- Rétroingénierie statique (+ intro à ghidra)
  - ▶ Algos de désassemblage (linéaire, récursif, Galileo, variantes)
  - ▶ Graphes de flots de contrôle (CFG)
  - ▶ Détection de fonctions (headers ELF + strip) : algo Nucleus
  - ▶ Def-use chain, dominateurs, boucles, boucles naturelles
  - ▶ Représentations intermédiaires
  - ▶ Typage (statique vs dynamique, flow vs value based, ...)

# Rappels de la dernière fois

- Rétroingénierie dynamique
  - ▶ Motivations, vue générale, méthode
  - ▶ Exemples d'outils (ptrace, gdb, wireshark)
  - ▶ Instrumentation de binaire (frida)
- Rétroingénierie concolique
  - ▶ Execution symbolique
  - ▶ Frameworks (angr, KLEE)
  - ▶ Fuzzing et approches (afl)
- Canaux auxiliaires
  - ▶ Exécution spéculative, caches, rowhammer, cold-boot attacks
  - ▶ Bruit/TEMPEST/Consommation électrique
  - ▶ Hardware trojans
- Injection de fautes
  - ▶ Voltage glitching/Clock skewing/EM Disturbance/Laser glitching/Heat attacks
  - ▶ Attaques + un peu de crypto (fault analysis).

# Au programme d'aujourd'hui

- En pratique, comment on fait du reverse?
- Quelques horreurs qu'on voit en reverse.
- Après midi : examen (un mélange d'un peu tout ce qu'on a vu).

<https://github.com/5d54626cc43ba7ce625903ffa39c049c/RV>

Note : pour l'examen tout est autorisé (sauf la triche sur son voisin) !

Vérifiez la présence sur votre machine :

- ghidra (ou autre outil de rétro-ingénierie statique)
- gdb/... (ou de quoi faire de la rétro-ingénierie dynamique)
- De quelques outils de base pour faire du reverse (binwalk, python, gcc, ...).

# Le reverse ne commence pas avec ghidra

Scénario typique :

- Homme politique random/manager random/... : "Bonjour, je vous ai ramené ça, vous pouvez le reverse ?"
- Vous : "Euh..."
- Homme politique random : "Merci !"
- Vous vous retrouvez avec un objet à reverser (généralement plusieurs exemplaires).

# Le reverse ne commence pas avec ghidra

Scénario typique :

- Homme politique random/manager random/... : "Bonjour, je vous ai ramené ça, vous pouvez le reverse ?"
- Vous : "Euh..."
- Homme politique random : "Merci !"
- Vous vous retrouvez avec un objet à reverser (généralement plusieurs exemplaires).

Exemple : ce magnifique routeur chinois.

# Étape 1 : l'ouvrir

- Avoir les bons outils (tournevis + pry spudgers)
- Parfois, c'est chiant (serrure, vis sécurité, ...)
- Parfois, c'est **très** chiant (serrure correcte, epoxy, étiquettes, scellés)
- Des fois, c'est plus facile de désamorcer une bombe que d'ouvrir cette m\* (contacteurs, capteurs de lumière, alarmes, détecteurs en tout genre ...)

## Étape 2 : identifier ce qu'on peut sur le circuit

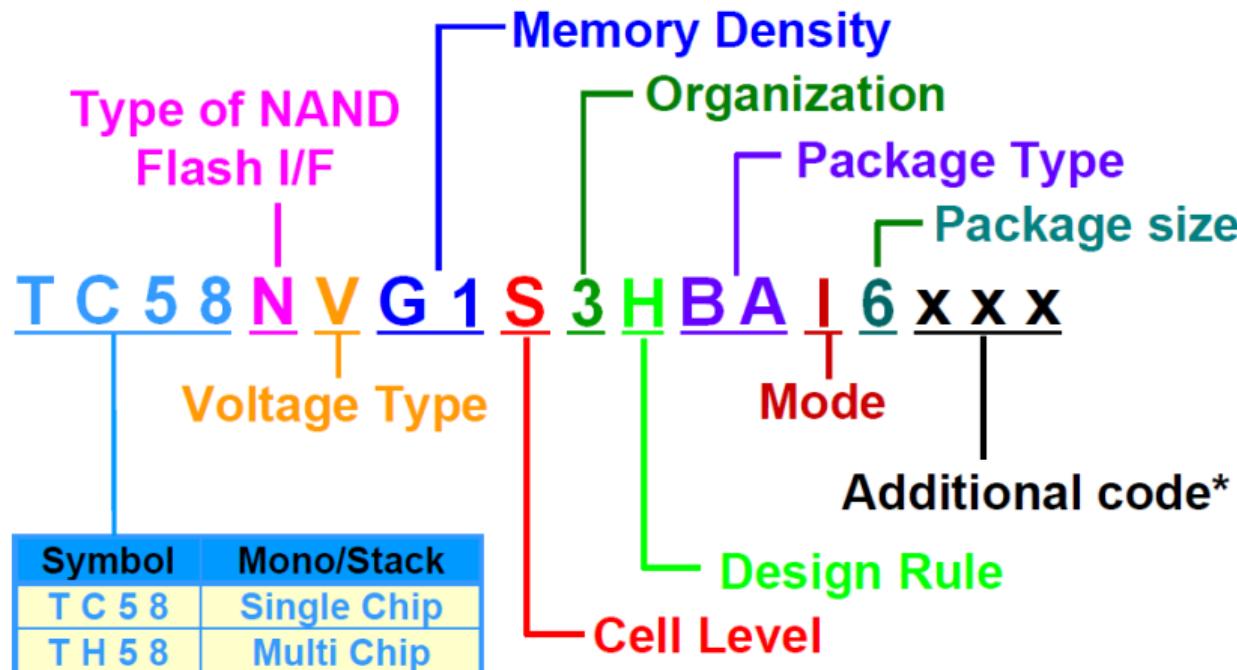
- Ce qu'on a déjà gratuitement : alimentation, ports, I/O
- Avec un voltmètre : trouver la masse, le 5V, où va quoi.
- Identifier les puces







## P/N Decoder for Raw NAND( Large Block ) ~Definition



This rule is available for 56nm, 43nm, 32nm and 24nm NANDs

T C 5 8 N V G 1 S 3 H B A I 6 x x x

Symbol	Vcc	VccQ
V	3.3V	-
Y	1.8V	-
A	3.3V	1.8V
B	3.3V	1.65V to 3.6V
D	3.3V or 1.8V	3.3V or 1.8V

Symbol	TYPE of NAND I/F
N	NAND
D	NAND <sup>1</sup>
T	Toggle mode NAND

\*1: Unique character for product variety control.

Symbol	Density
M8	256( $=2^8$ ) Mbits =32MB
M9	512( $=2^9$ ) Mbits =64MB
G0	1( $=2^0$ ) Gbits =128MB
G1	2( $=2^1$ ) Gbits =256MB
G2	4( $=2^2$ ) Gbits =512MB
G3	8( $=2^3$ ) Gbits =1GB
G4	16( $=2^4$ ) Gbits =2GB
GA	24 Gbits =3GB
G5	32( $=2^5$ ) Gbits =4GB
GB	48 Gbits =6GB
G6	64( $=2^6$ ) Gbits =8GB
GC	96 Gbits =12GB
G7	128( $=2^7$ ) Gbits =16GB
GD	192 Gbits =24GB
G8	256( $=2^8$ ) Gbits =32GB
GE	384 Gbits =48GB
G9	512( $=2^9$ ) Gbits =64GB
GF	768 Gbits =96GB
T0	1( $=2^0$ ) Tbits =128GB
T1	2( $=2^1$ ) Tbits =256GB

## P/N Decoder for Raw NAND( Large Block ) ~Details-2

This rule is available for 56nm, 43nm, 32nm and 24nm NANDs

T C 5 8 N V G 1 S 3 H B A I 6 X X X																																							
<table border="1"><thead><tr><th>Symbol</th><th>Cell Level</th></tr></thead><tbody><tr><td>S / H<sup>*1</sup></td><td>2 Level( 1 bits/cell )</td></tr><tr><td>D / E<sup>*1</sup></td><td>4 Level( 2 bits/cell )</td></tr><tr><td>T / U<sup>*1</sup></td><td>8 Level( 3 bits/cell )</td></tr></tbody></table>	Symbol	Cell Level	S / H <sup>*1</sup>	2 Level( 1 bits/cell )	D / E <sup>*1</sup>	4 Level( 2 bits/cell )	T / U <sup>*1</sup>	8 Level( 3 bits/cell )	<table border="1"><thead><tr><th>Symbol</th><th>x8</th><th>x16</th><th>Page Size</th><th>Block Size</th></tr></thead><tbody><tr><td>0</td><td>5</td><td></td><td>4KB</td><td>256KB</td></tr><tr><td>1</td><td>6</td><td></td><td>4KB</td><td>512KB</td></tr><tr><td>2</td><td>7</td><td></td><td>&gt;4KB</td><td>&gt;512KB</td></tr><tr><td>3</td><td>8</td><td></td><td>2KB</td><td>128KB</td></tr><tr><td>4</td><td>9</td><td></td><td>2KB</td><td>256KB</td></tr></tbody></table>	Symbol	x8	x16	Page Size	Block Size	0	5		4KB	256KB	1	6		4KB	512KB	2	7		>4KB	>512KB	3	8		2KB	128KB	4	9		2KB	256KB
Symbol	Cell Level																																						
S / H <sup>*1</sup>	2 Level( 1 bits/cell )																																						
D / E <sup>*1</sup>	4 Level( 2 bits/cell )																																						
T / U <sup>*1</sup>	8 Level( 3 bits/cell )																																						
Symbol	x8	x16	Page Size	Block Size																																			
0	5		4KB	256KB																																			
1	6		4KB	512KB																																			
2	7		>4KB	>512KB																																			
3	8		2KB	128KB																																			
4	9		2KB	256KB																																			
<p>*1: Unique character for product variety control.</p>	<table border="1"><thead><tr><th>Symbol</th><th>Design Rule</th></tr></thead><tbody><tr><td>A</td><td>130 nm</td></tr><tr><td>B</td><td>90 nm</td></tr><tr><td>C</td><td>70 nm</td></tr><tr><td>D</td><td>56 nm</td></tr><tr><td>E</td><td>43 nm</td></tr><tr><td>F</td><td>32nm</td></tr><tr><td>G</td><td>24nm A-type</td></tr><tr><td>H</td><td>24nm B-type</td></tr></tbody></table>	Symbol	Design Rule	A	130 nm	B	90 nm	C	70 nm	D	56 nm	E	43 nm	F	32nm	G	24nm A-type	H	24nm B-type																				
Symbol	Design Rule																																						
A	130 nm																																						
B	90 nm																																						
C	70 nm																																						
D	56 nm																																						
E	43 nm																																						
F	32nm																																						
G	24nm A-type																																						
H	24nm B-type																																						

## P/N Decoder for Raw NAND( Large Block ) ~Details-3

This rule is available for 56nm, 43nm, 32nm and 24nm NANDs

T C 5 8 N V G 1 S 3 H B A | 6 x x x

PKG	Symbol	Lead Free	Halogen Free
TSOP	FT	No	No
	TG <sup>1</sup>	Yes	No
	TA	Yes	Yes
BGA	XB	No	No
	XG <sup>1</sup>	Yes	No
	BA	Yes	Yes
LGA	-	No	No
	XL <sup>1</sup>	Yes	No
	LA	Yes	Yes

Normal	I-ver.	Symbol	Channel	# of /CE	Note
0	I	Single	1	TSOP, BGA	
2	K	Single	2	TSOP, BGA	
4	M	Dual	2	LGA	
7	R	Single	4	TSOP PoP	
8	S	Single / Dual	4	TSOP, BGA,LGA	
A	U	Single / Dual	6	TSOP, BGA,LGA	
B	V	Single / Dual	8	TSOP, BGA,LGA	

<sup>1</sup>: Some of the product are Halogen Free with this code. If necessary, Please ask to Toshiba.

# Tiré du site du constructeur

This rule is available for 56nm, 43nm, 32nm and 24nm NANDs

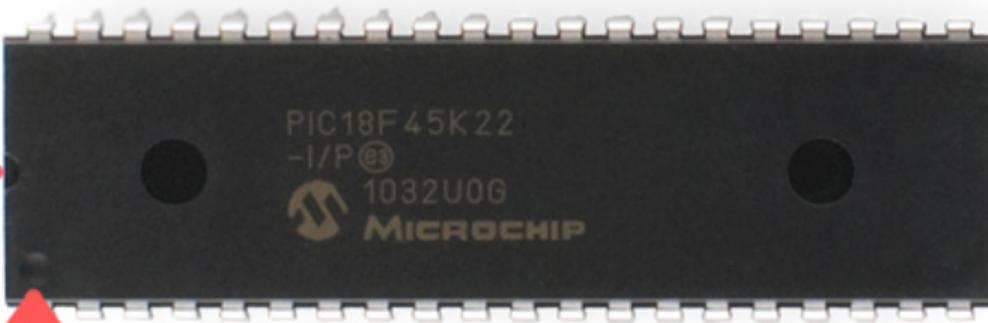
**T C 5 8 N V G 1 S 3 H B A I 6 x x x**

Symbol	TSOP [mm]	LGA [mm]	BGA [mm]
0	12 x 20 x 1.2	Reserved	General code( 12.5x20, 14x18, 12x18 )
1	Reserved	40 lands, 12 x 18 x 0.7	224 balls, 14x18x1.46 <sup>-1</sup>
2	Reserved	40 lands, 12 x 18 x 1.15	224 balls, 14x18x1.46 <sup>-1</sup>
3	Reserved	40 lands, 12 x 17 x 0.65	60 balls, 8.5 x 13
4	Reserved	40 lands, 12 x 17 x 1.0	60 balls, 9 x 11
5	Reserved	40 lands, 12 x 17 x 1.04	60 balls, 10 x 13
6	Reserved	40 lands, 13 x 17 x 1.04	60 balls, 8.5 x 13 <sup>-1</sup>
7	Reserved	52 lands, 14 x 18 x 1.4	60 balls, 9 x 11 <sup>-1</sup>
8	Reserved	52 lands, 14 x 18 x 1.04	60 balls, 10 x 13 <sup>-1</sup>
9	Reserved	52 lands, 14 x 18 x 1.0	132 balls( Toggle ), 12x18x1.4
A	Reserved	52 lands, 12 x 17 x 1.04/1.0	132 balls( Toggle ), 12x18x1.85
B	12 x 18 x 1.2 <sup>-1</sup>	52 lands, 12 x 17 x 1.4	224 balls, 14x18x1.35
C	Reserved	52 lands, 11x14x0.9	-
D	Reserved	132 lands( Toggle ) 12x18x1.04	-

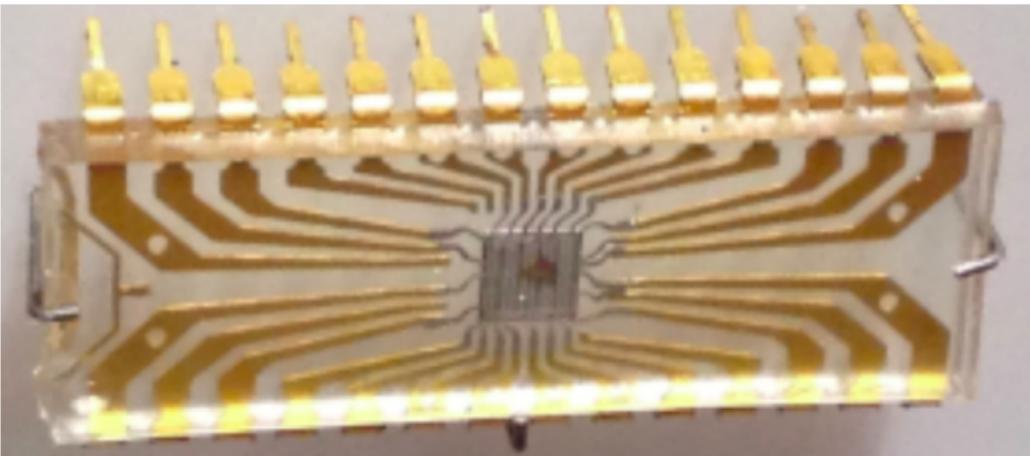
\*1: Unique character for product variety control.



Notch →



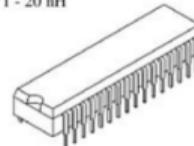
↑  
Dot



- DIL (Dual In Line)

- Low pin count
- Large

Package inductance:  
1 - 20 nH

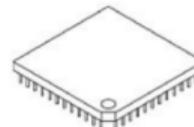


- PGA (Pin Grid Array)

- High pin count (up to 400)
- Previously used for most CPU's

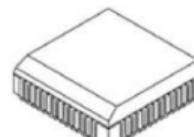
- PLCC (Plastic leaded chip carrier)

- Limited pin count (max 84)
- Large
- Cheap
- SMD



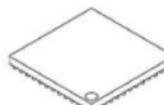
- QFP (Quarter Flat pack)

- High pin count (up to 300)
- small
- Cheap
- SMD



- BGA (Ball Grid Array)

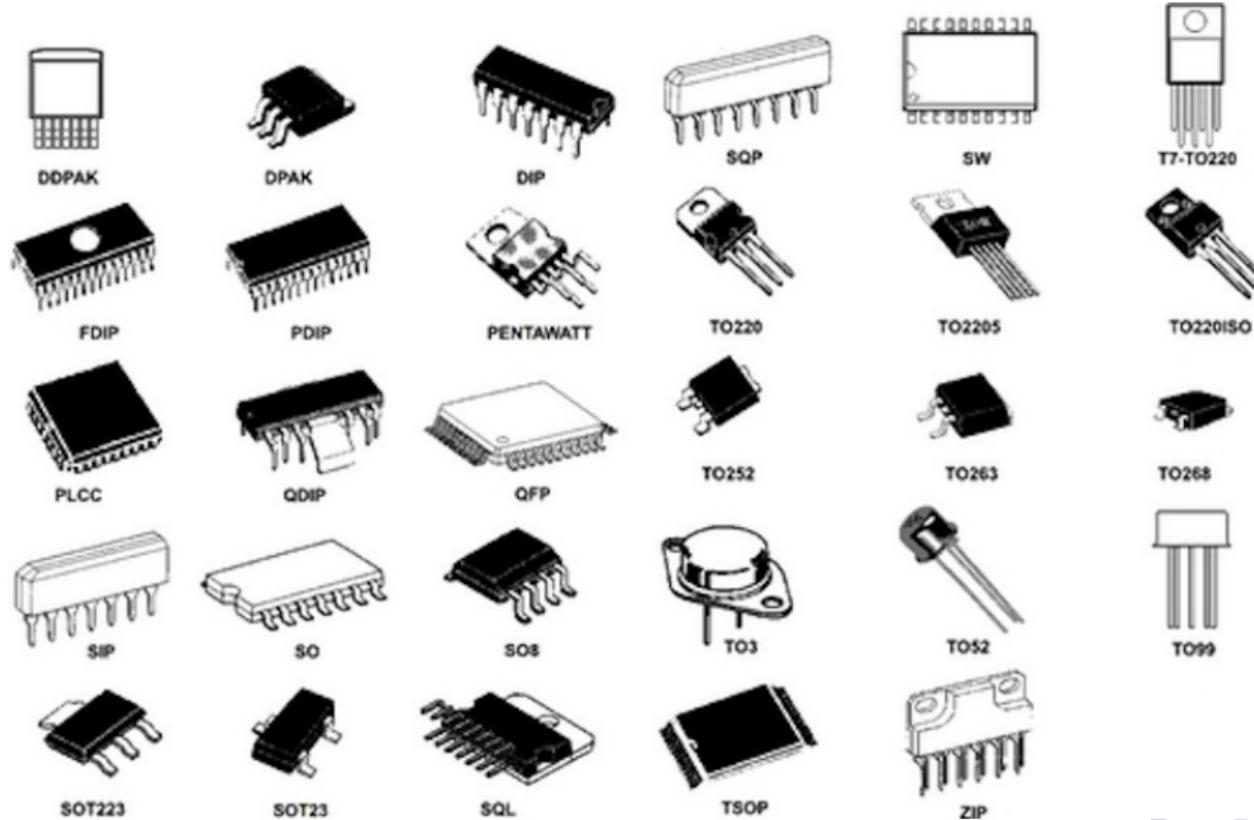
- Small solder balls to connect to board
- small
- High pin count
- Cheap
- Low inductance



Package inductance:  
1 - 5 nH



# Plein de packaging (pour le reste, Wikipédia)



## Et ensuite

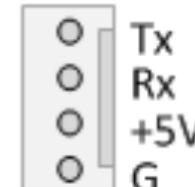
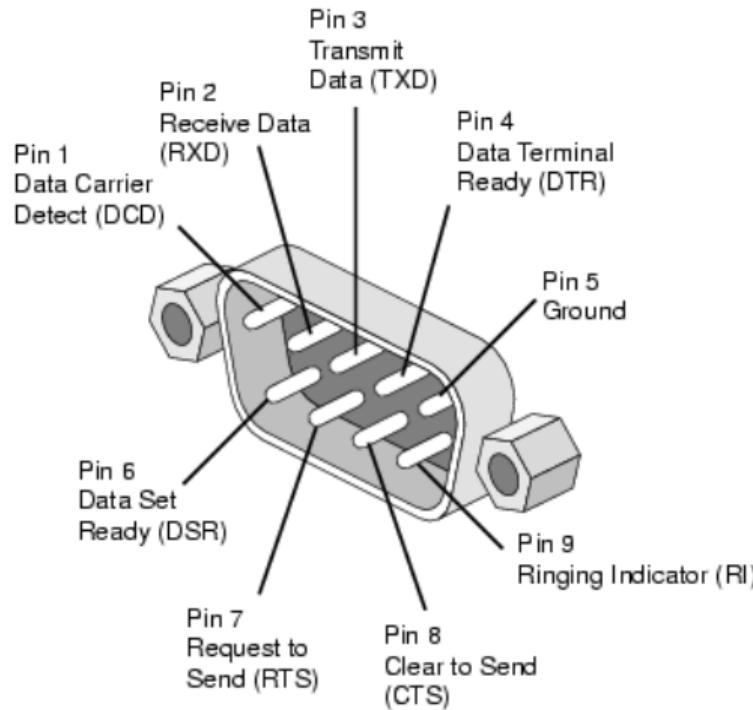
- Avec de la chance, on a le numéro du modèle, et on récupère la datasheet.
- Avec de la chance, la datasheet du SoC contient la memory map.
- Avec beaucoup de chance, tout est documenté.
- Sinon, on fait comme on peut.
  - ▶ Rechercher la puce à partir des pins identifiés (masse, power, jtag, ...)
  - ▶ Contacter l'usine chinoise "hello I want to buy 10k of your chips for my new board, can you send me the datasheet to be sure that it is compatible ?"
  - ▶ Rechercher du code lié au chip.

*Des indications fonctionnelles supplémentaires sont souvent nécessaires pour re-verser efficacement un firmware.*

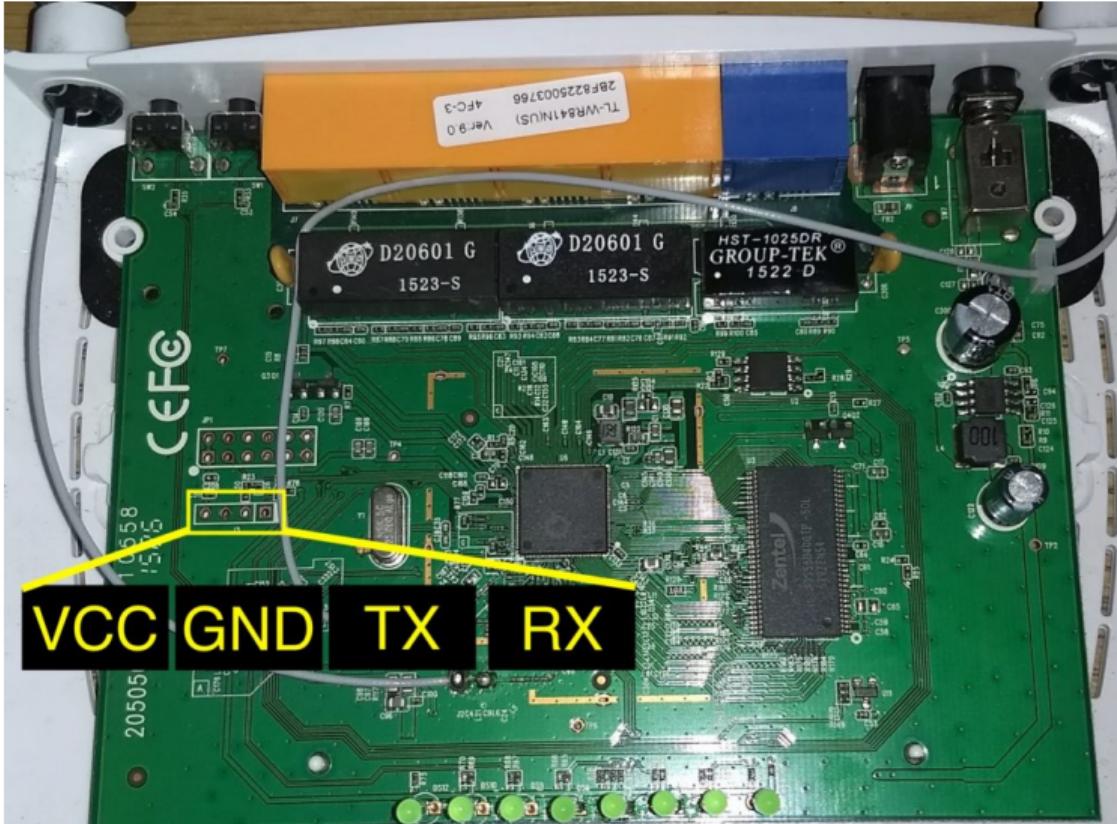
# La suite ? Les classiques

- UART
- SPI
- JTAG

# UART

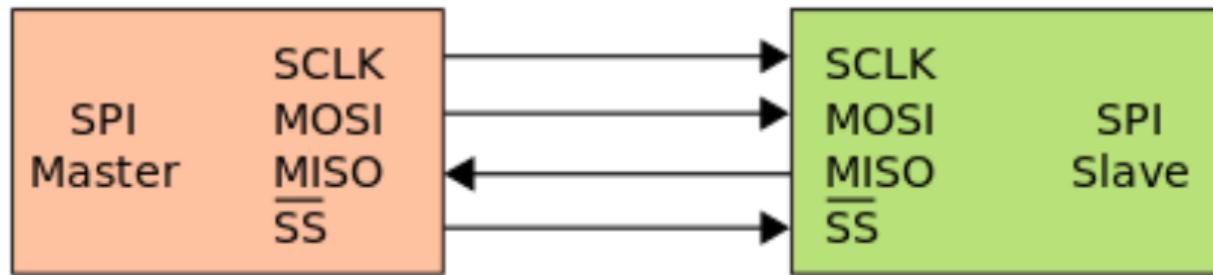


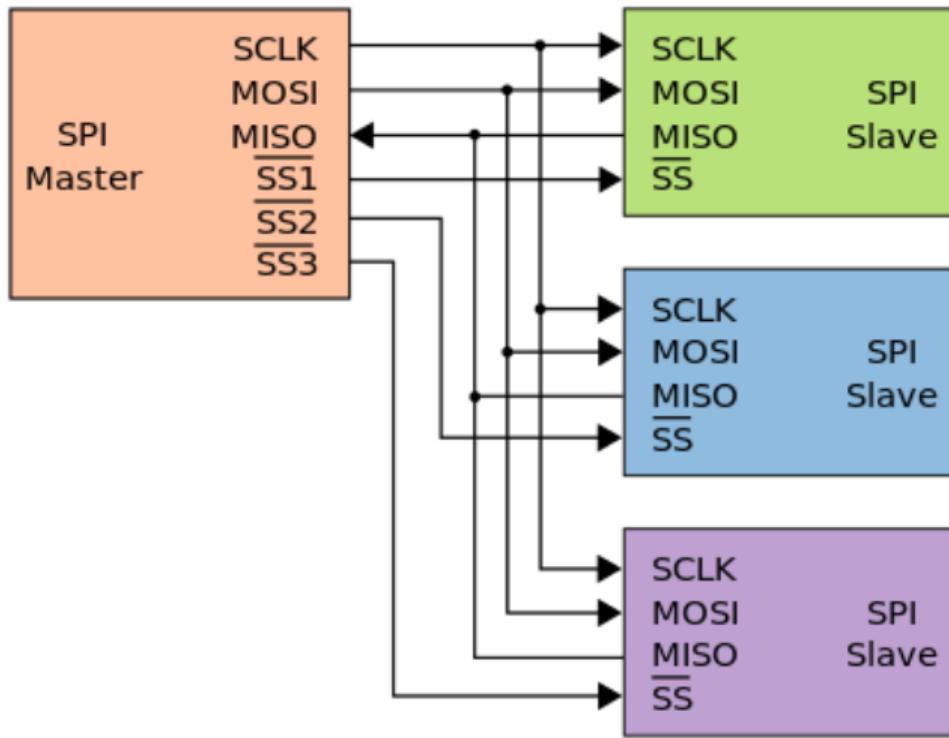
# UART



# UART demo

Demo (dumper la flash).





# SPI demo

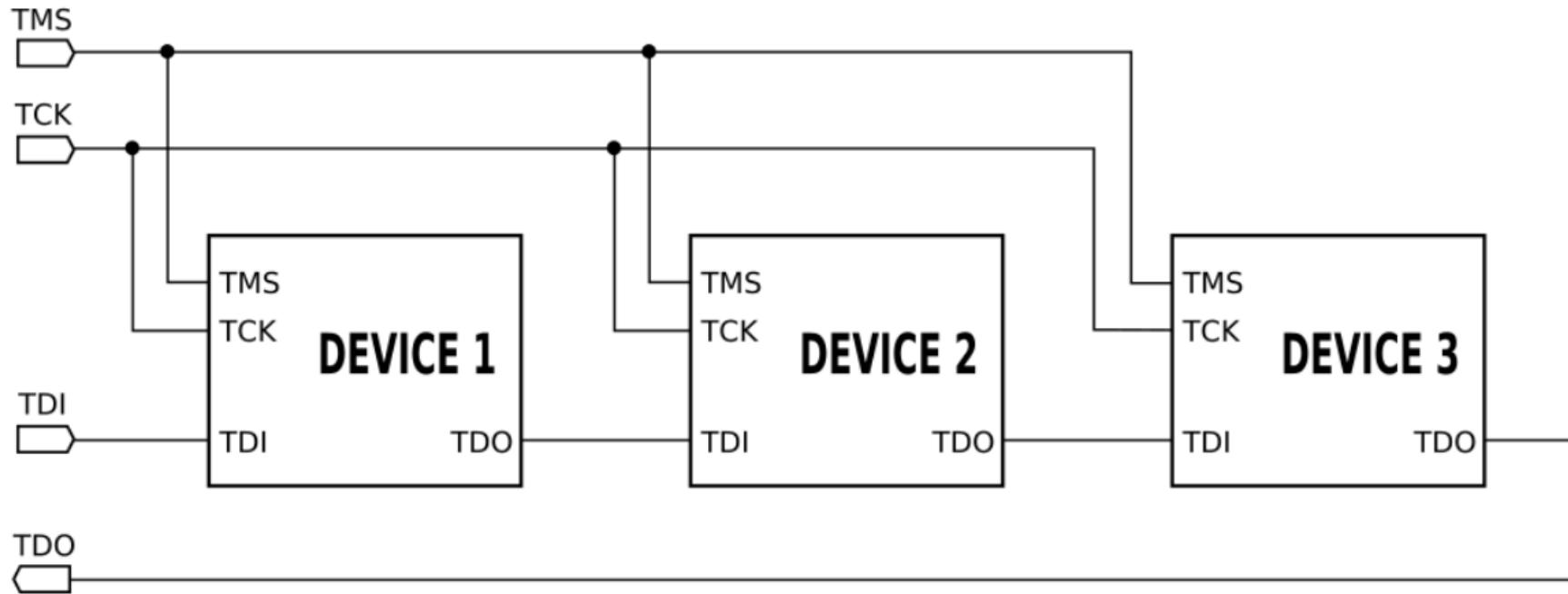
Malheureusement, Amazon n'a toujours pas livré :



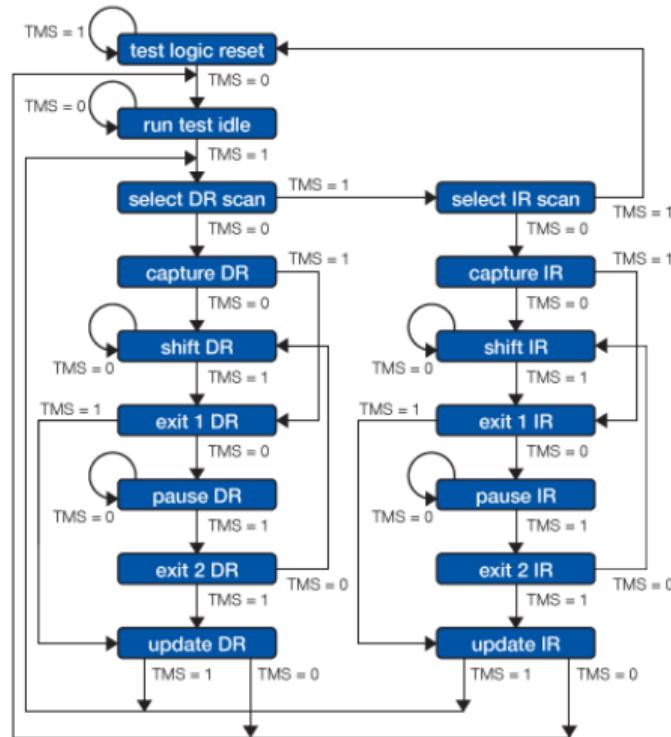
Du coup à faire chez soi : **flashrom** avec n'importe quelle RPI.

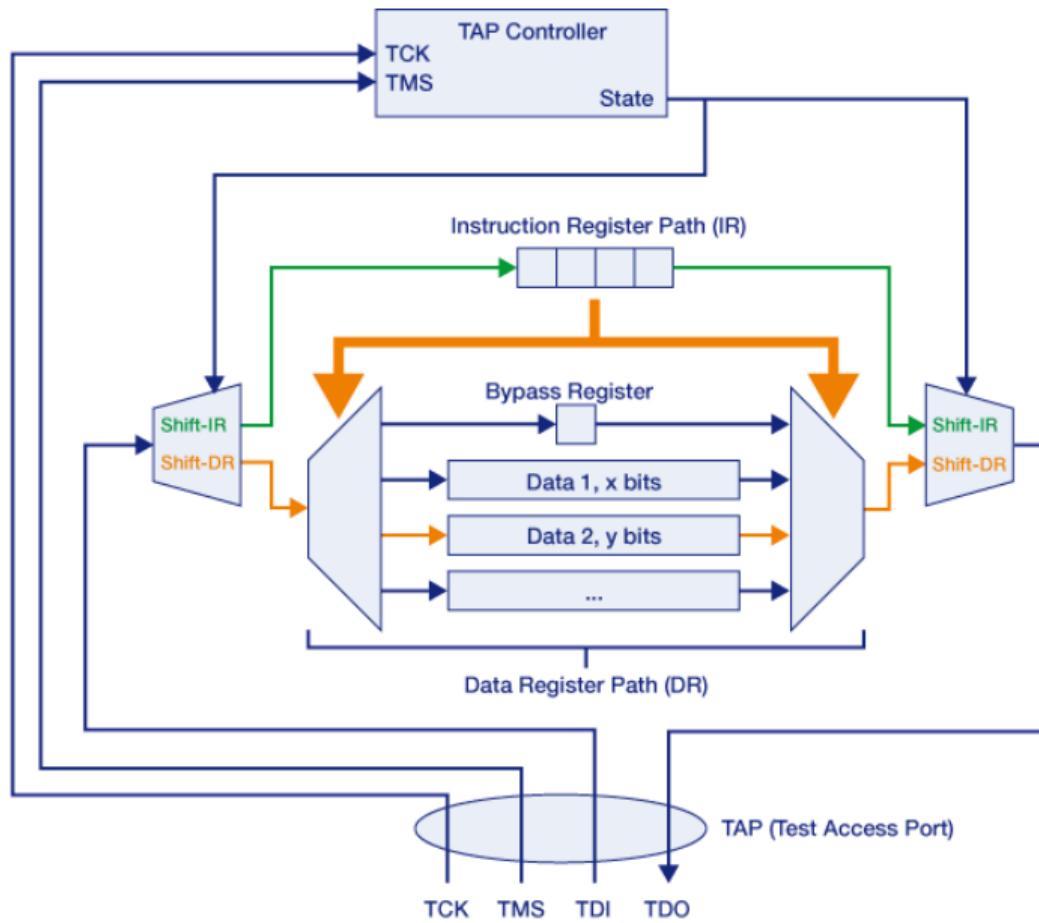
- Joint Test Action Group.
- Permet le debug des circuits.
- Pile de protocoles (comme pour OSI).

# Test Action Port and Boundary Scan (IEEE 1149.1)



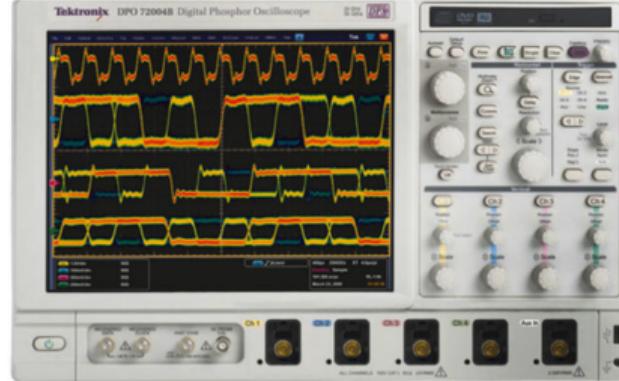
# TAP FSM





# TAP instructions

RTFM.



# On a un dump, quoi faire ensuite?

Demo binwalk.

# Pause (15 minutes)

Des questions ?

## Definition (Obfuscation)

L'obfuscation (ou offuscation) est un ensemble de méthodes permettant de transformer un programme en un autre dont la rétro-ingénierie est plus difficile.

### Pourquoi obfusquer ?

- Perte de temps
- Dissuader
- Méchanceté

### Qui obfusque ?

- Malware
- DRM
- Produits closed-source
- Données sensibles à protéger

# Catégorisation des obfuscations

Ninon Eyrolles, *Obfuscation par expressions mixtes arithmético-booleennes*, 2017

Propriétés de l'obfuscation :

- Même fonctionnalité
- Efficacité comparable (temps, mémoire, longueur de code)
- Résilience : le programme doit être plus dur à analyser

Trois grandes catégories d'obfuscation :

- Obfuscations au niveau du code source
- Obfuscations de flot de contrôle
- Obfuscations de données

# International Obfuscated C Code contest

```
char*I="ustvrttsuqqqqqqqqyyyyyyyyy }{|~z|{}"
" ____76Lsabcdcba_. pknbrq_-PKNBRQ_-?A6J57IKJT576,+ -48HLSU" ;
#define F getchar()&z
#define v X(0,0,0,21,
#define Z while(
#define _ ; if(
#define P return—G,y^=8,
B,i,y,u,b,I[411],*G=I,x=10,z=15,M=1e4;X(w,c,h,e,S,s){ int t,o,L,E,d,O=e,N=M*M,K
=78-h<<x,p,*g,n,*m,A,q,r,C,J,a=y?-x:x;y^=8;G++;d=w||s&&s>=h&&v 0,0)>M;do{ _ o=I[ p=O]) { q=o&z^y - q<7){A=q --&2?8:4;C=o-9&z?q["&..$.."] : 42; do{ r=I[p+=C[I]-64] - !w|p
==w){ g=q | p+a-S?0: I+S - !r &(q | A<3 || g) || ( r+1&z^y)>9&&q | A>2){ _ m=!(r-2&7))P G[1]=O.
K; J=n=o&z; E=I[p-a]&z; t=q | E-7?n:( n+=2,6^y); Z n<=t){ L=r ? I[r&7]*9-189-h-q:0 - s)L
+= (1-q ? I[p/x+5]-I[O/x+5]+I[p%x+6]*~-!q-I[O%x+6]+o/16*8:!!m*9)+(q ? 0:( I[p-1]^n)+
!( I[p+1]^n)+I[n&7]*9-386+!!g*99+(A<2))+!(E^y^9) - s>h||1< s&s==h&&L>z | d){ p[I]=n,O
[I]=m?*g==*m,*m=0:g?*g=0:0;L=X(s>h | d?0:p,L-N,h+1,G[1],J=q | A>1?0:p,s)-!(h || s-1|B
-O| i-n | p-b | L<-M))P y^=8,u=J; J=q-1|A<7||m||!s | d | r | o<z || v 0,0)>M;O[I]=o;p[I]=r;m?
*m==*g,*g=0:g?*g=9^y:0; } - L>N){ *G=O - s>1){ _ h&&c-L<0)P L - !h)i=n,B=O,b=p;} N=L; }
n+=J || (g=I+p,m=p<O?g-3:g+2,*m<z | m[O-p] || I[p+=p-O]); } })} Z!r&q>2||(p=O,q | A>2|o>z&
!r&&++C*-A)); } }) Z+=O>98?O=20:e-O); P N+M*M&&N>K+1924|d?N:0; } main(){ Z+B<121)*G
++=B/x%o<2|B%o<2?7:B/x&4?0:* I++&31;Z B=19){ Z B++<99)putchar(B%o?I[B[I]|16]:x) -
x-(B=F)){ i=I[B+=(x-F)*x]&z; b=F; b+=(x-F)*x; Z x-(*G=F)) i=*G^8^y; } else v u,5); v u,
1); } }
```

# International Obfuscated C Code contest

Petite démo.

# Pour la culture

Le record du plus petit jeu d'échec en C a été battu en 2014 (Super Micro Chess) :

```
#define F; if(
#define W while(
C=799,K=8,X,Y; char c[9],b[128]="VSUWTUSV";D(k,x,n){int i=0,j,t,p,
u,r,y,m=C,v;do{F(u==b[i])&k){j=".H?LFICF"[p=r=u&7]-64;W r=p>2&
r<0?-r:64-01/@AP@ABPOQ@NR_a@'[++j)){y=i;do{t=b[y+=r]F(p==7|x
)&&j==8||!(r&7)-!t&p<3|t&k||y&136)break;v=t&k?1:"_!!#~#%)"[t
&7]-32 F n&&v<64){b[i]=0,b[y]=u F p<3&&y+r+1&128)b[y]=(*c&c
[4]?c[4]:55)-48|k,v+=9;v=D(24-k,2,n-1)F x&1&v>-64&i==X&y==Y){
F j==8)b[y+(r>>2^1)]=0,b[y-r/2]=6|k;return 0;}b[i]=u,b[y]=t;}F
v>m){m=v F n>4)X=i,Y=y; } t+=p<5 F x&1&&(y&112)+6*k==128&p<3)t
--;}W!t);}}W i=i+9&119);return m;}main(){X=8;W X--)b[X+112]=(
b[X]==64)-8,b[X+16]=18,b[X+96]=9;W 1){X=128;W X--)putchar(X
&8&&(X==7)?10:".?+nkbrq?*?NKBRQ"[b[X]&15]);gets(c);X=*(c-16*c
[1]+C,Y=c[2]-16*c[3]+C F!*c)D(K,0,5)F!D(K,1,1))K^=24;}}}
```

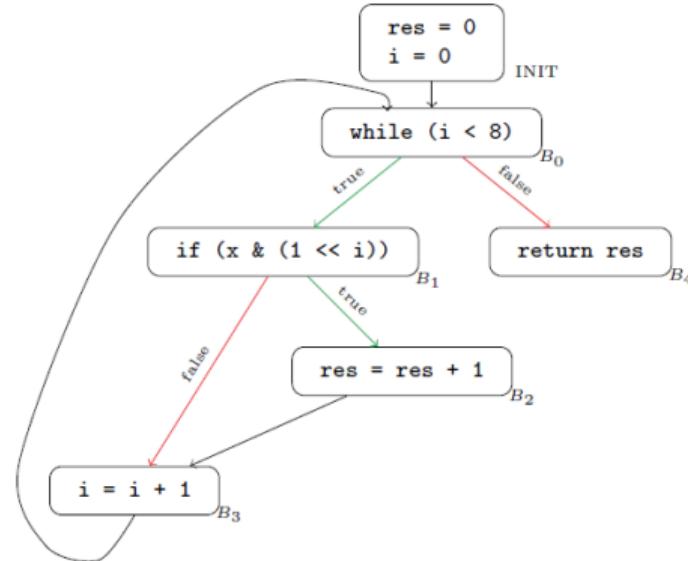
# Obfuscations de flot de contrôle

- ABI custom.
- Inlining.
- volatile keyword pour casser les optimisations du compilateur.
- Prédicats opaques : grand théorème de Fermat,  $\forall n > 2, \nexists x, y, z \in \mathbb{N}, x^n + y^n = z^n$
- Control-flow flattening (voire virtualisation)

# CFG flattening

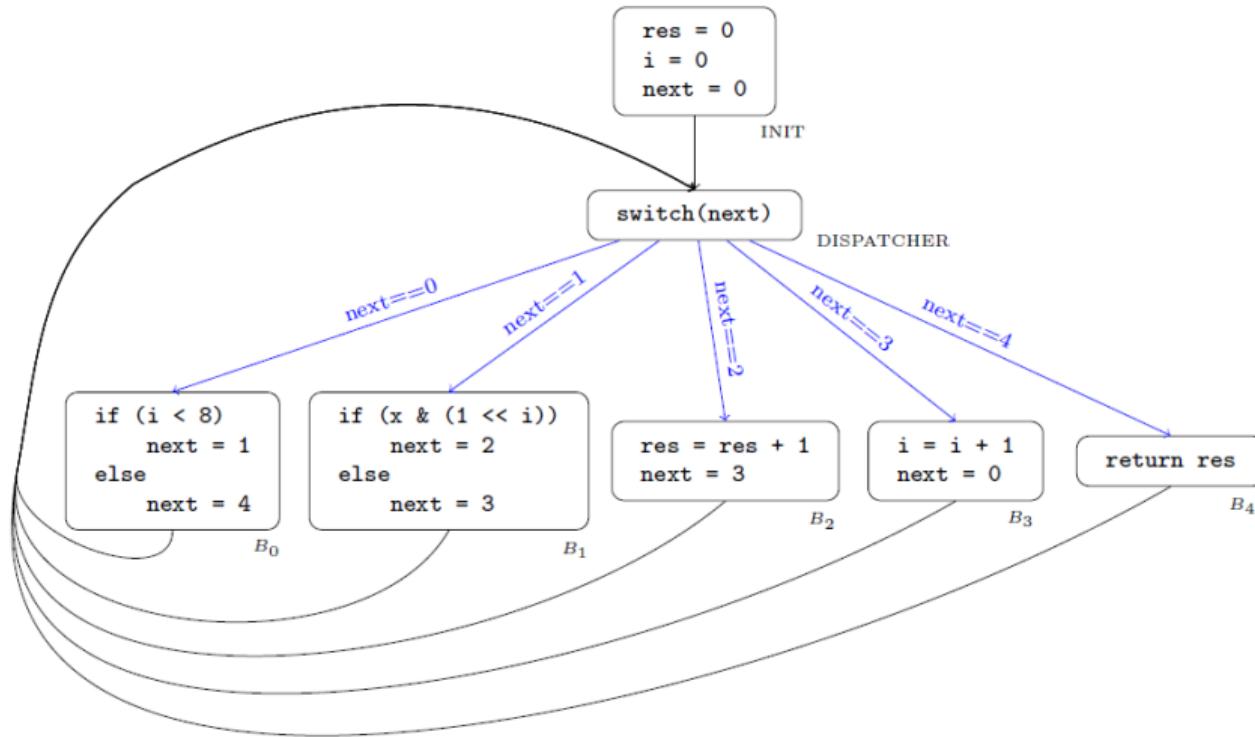
```
uint8_t hamming_weight(uint8_t x)
{
    uint8_t res = 0;
    uint8_t i = 0;
    while(i < 8) {
        if (x & (1 << i))
            res = res + 1;
        i = i + 1;
    }
    return res;
}
```

(a) Function in C.



(b) CFG in pseudo-code.

# CFG flattening



# Obfuscation de données

- Séparation de variables  $x = q \times d + r$  et ensuite  $x := x + 1 \rightarrow \begin{cases} q := q + \lfloor \frac{r}{d-1} \rfloor \\ r := r + 1 \bmod d \end{cases}$
- Changement de base (base 3 ou base 64 par exemple) ou d'encodage (look-up table, wide-chars : strings -e L).
- Masquage des données  $x \rightarrow x \oplus a$
- MBA (mixed boolean arithmetics) :  $x + y \rightarrow (x \oplus y) + 2 \times (x \wedge y)$
- Insertions d'identités : par exemple  $x \rightarrow (x \times 39 + 23) \times 151 + 111$  (8 bits)
- Constantes opaques : polynômes inversibles dans  $\mathbb{Z}/2^n\mathbb{Z}$ ,  $x \rightarrow P(E + Q(cste))$  ( $E$  expression égale à 0 en MBA)

# Protections anti-reverse

- Unpacking
- Chiffrement du payload
- Virtualisation
- Polymorphisme
- Canaries

# Détection de la rétroingénierie

Collin Mulliner, *Detecting Reverse Engineering with Canaries*

Analyse statique vs analyse dynamique :

- Désassemblage / Binary diffing / Fingerprinting
- Lent et couteux
- Résultats plus complets que corrects.
- Indétectable
- Déboggage / Instrumentation / Fuzzing / Traces (systèmes, réseaux, ...)
- Rapide, peu couteux
- Résultats plus corrects au détriment de la complétude.
- DéTECTABLE

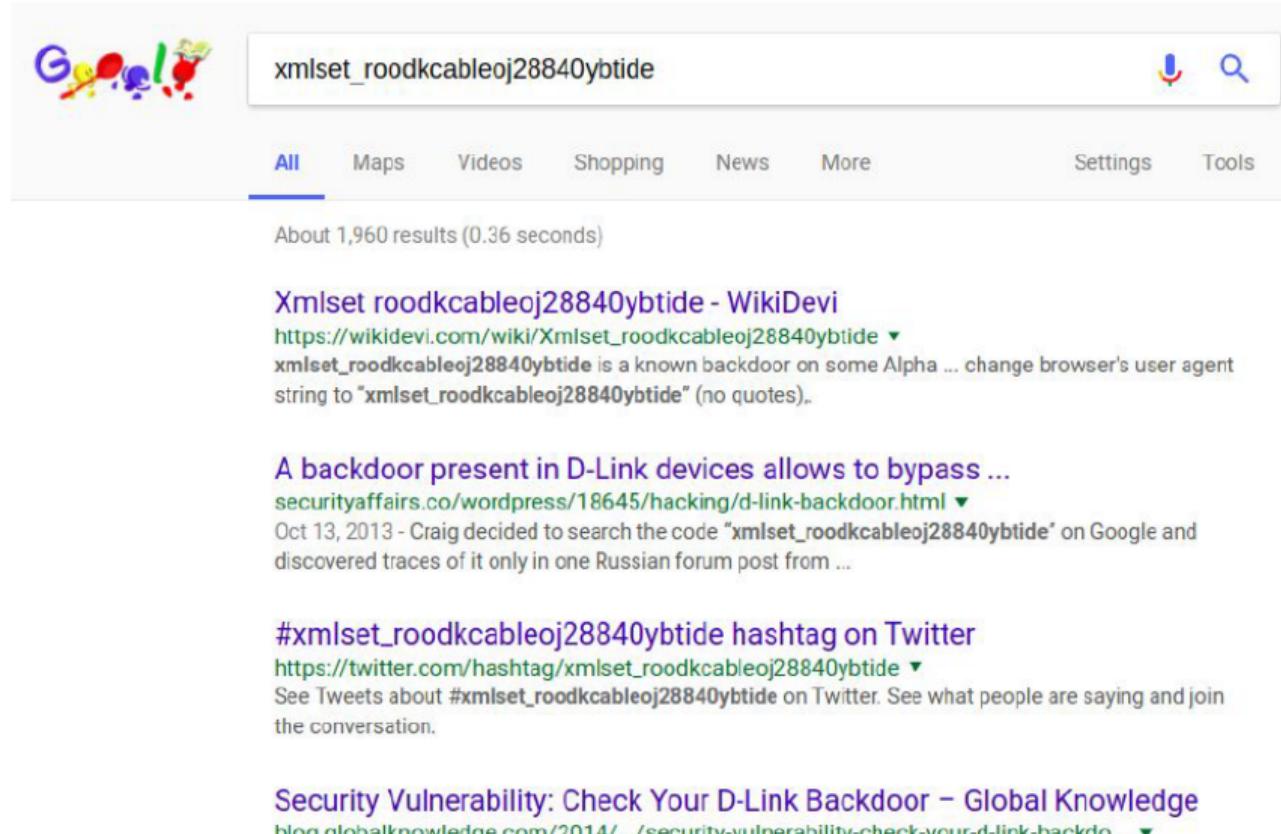
Analyse statique bien meilleure. Mais lente et couteuse ⇒ économies de bout de chandelle.

# Un exemple

```
lw      $a0, 0xB8($s0)
la      $t9, strstr
nop
jalr    $t9 ; strstr
nop
lw      $gp, 0x3B8+saved_gp($sp)
nop
la      $a1, 0x470000
nop
addiu   $a1, ($aXmlset_roodk_0 - 0x470000)  # "xmlset_roodkcableoj28840ybtide"
bnez    $v0, end
li      $v1, 1
```

```
lw      $a0, 0xD0($s0)
la      $t9, strcmp
nop
jalr    $t9 ; strcmp
nop
lw      $gp, 0x3B8+saved_gp($sp)
beqz   $v0, end
li      $v1, 1
```

# Un exemple



A screenshot of a Google search results page. The search query in the bar is "xmlset\_roodkcableoj28840ybtide". The results are as follows:

- Xmlset roodkcableoj28840ybtide - WikiDevi**  
[https://wikidevi.com/wiki/Xmlset\\_roodkcableoj28840ybtide](https://wikidevi.com/wiki/Xmlset_roodkcableoj28840ybtide) ▾  
xmlset\_roodkcableoj28840ybtide is a known backdoor on some Alpha ... change browser's user agent string to "xmlset\_roodkcableoj28840ybtide" (no quotes).
- A backdoor present in D-Link devices allows to bypass ...**  
<https://securityaffairs.co/wordpress/18645/hacking/d-link-backdoor.html> ▾  
Oct 13, 2013 - Craig decided to search the code "xmlset\_roodkcableoj28840ybtide" on Google and discovered traces of it only in one Russian forum post from ...
- #xmlset\_roodkcableoj28840ybtide hashtag on Twitter**  
[https://twitter.com/hashtag/xmlset\\_roodkcableoj28840ybtide](https://twitter.com/hashtag/xmlset_roodkcableoj28840ybtide) ▾  
See Tweets about #xmlset\_roodkcableoj28840ybtide on Twitter. See what people are saying and join the conversation.
- Security Vulnerability: Check Your D-Link Backdoor – Global Knowledge**  
[blog.globalknowledge.com/2014/.../security-vulnerability-check-your-d-link-backdo...](https://blog.globalknowledge.com/2014/.../security-vulnerability-check-your-d-link-backdo...) ▾

# Reverse canaries

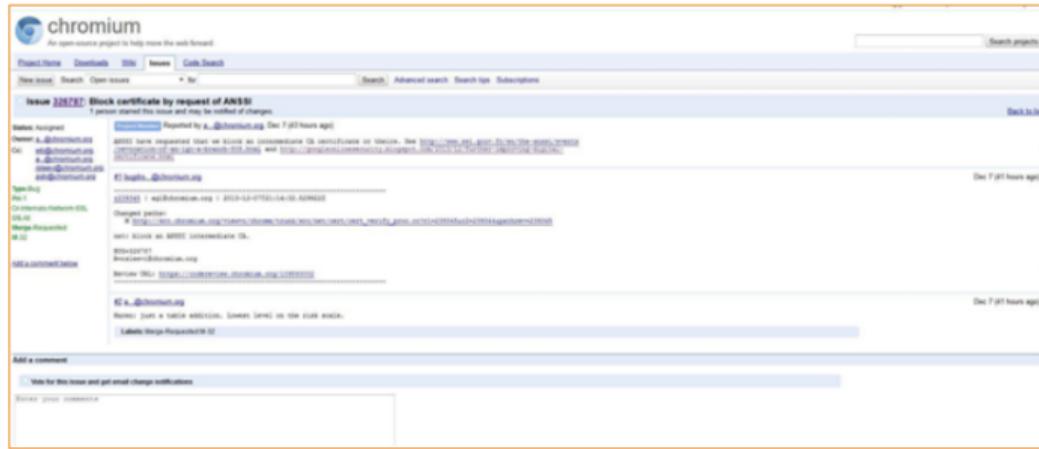
- Recherche google d'une chaîne très spécifique, avec souvent zéro ou un seul résultat.
- Le reverseur va quasi-certainement cliquer.
- Que se passe-t-il si le site est un piège ? Le fabricant reçoit une notification de reverse (avec de la chance, IP du reverseur).

# Reverse canaries

- Recherche google d'une chaîne très spécifique, avec souvent zéro ou un seul résultat.
- Le reverseur va quasi-certainement cliquer.
- Que se passe-t-il si le site est un piège ? Le fabricant reçoit une notification de reverse (avec de la chance, IP du reverseur).
- Variantes :
  - ▶ API endpoint [https://api.site.net/service/auth\\_no\\_password](https://api.site.net/service/auth_no_password)
  - ▶ DNS resolution : <https://magcicanary3421234.site.net>
  - ▶ Hard-coded fake credentials (mobile apps, AWS, ...)
  - ▶ Fichier PDF/docx/... piégé
- Poussons le vice : enregistrer le canary sur Adwords, notification automatique si recherche google effectuée contenant le canary.

# Petite histoire

Google a publié un bulletin pour dénoncer un problème de certificat apparu sur leurs écrans le 3 décembre. Une enquête a mis en évidence plusieurs certificats non autorisés visant plusieurs domaines Google et émis par une autorité de certification en relation avec l'ANSSI.



Pour colmater la brèche, Google a bloqué ces certificats corrompus dans Chrome, tout en entrant en discussions avec l'ANSSI. Selon les éléments en possession du géant américain, ces certificats étaient utilisés sur le réseau privé d'un dispositif commercial, en pleine connaissance des utilisateurs afin d'inspecter le trafic chiffré.

## Une violation grave



# Antidebug

- IsDebuggerPresent dans Debugapi.h (Windows)

# Antidebug

- IsDebuggerPresent dans Debugapi.h (Windows)

```
DWORD64 dwpeb = __readgsqword(0x60);  
*((PBYTE)(dwpeb + 2)) = 0;
```

# Antidebug

- IsDebuggerPresent dans Debugapi.h (Windows)

```
DWORD64 dwpeb = __readgsqword(0x60);  
*((PBYTE)(dwpeb + 2)) = 0;
```

- Variante pour Windows NT, NtGlobalFlag à un autre endroit dans la même structure.
- Trap Flag TF (x86) : permet de faire du single step. Le processeur lance une exception INT 01h à chaque instruction.

```
--try  
{  
    __asm  
    {  
        pushfd  
        or dword ptr[esp], 0x100 // set the Trap Flag  
        popfd                  // Load the value into EFLAGS  
        nop  
    }  
}  
__except (EXCEPTION_EXECUTE_HANDLER)  
// debugger is not present
```

- CheckRemoteDebuggerPresent(GetCurrentProcess(), &myboolean )
- Se débugger soi-même : impossible d'attacher deux debuggers au même processus via DebugActiveProcess.
- Calculer dynamiquement un hash d'une fonction : détection de hooking/binary patching.
- Timings : calculer le temps que prend une fonction

Résumé plus détaillé sur :

<http://antukh.com/blog/2015/01/19/malware-techniques-cheat-sheet/>  
+ google pour retrouver les morceaux de code pour les contourner.

S'il reste du temps.

Fin

Questions ?