IpMorph :

« vers la mystification de la prise d'empreinte de pile »

Présentation à l'OSSIRB
17 février 2009

http://dev.ipmorph.org

IpMorph : « vers la mystification de la prise d'empreinte de pile »

Sommaire

- Contexte
- Historique
- Etymologie
- Techniques de détection
- Etat de l'art de la mystification
- Utilisation d'IpMorph
- IpMorph != Packet Purgatory / Morph
- Socle logiciel
- Architecture générale
- Translation ARP
- Mystification de la congestion
- Focus : Calcul ISN
- Focus : Interfaces réseaux « read/write »
- Personality Manager
- Démonstration
- Etat d'avancement
- Perspectives
- Bibliographie

IpMorph : « vers la mystification de la prise d'empreinte de pile »

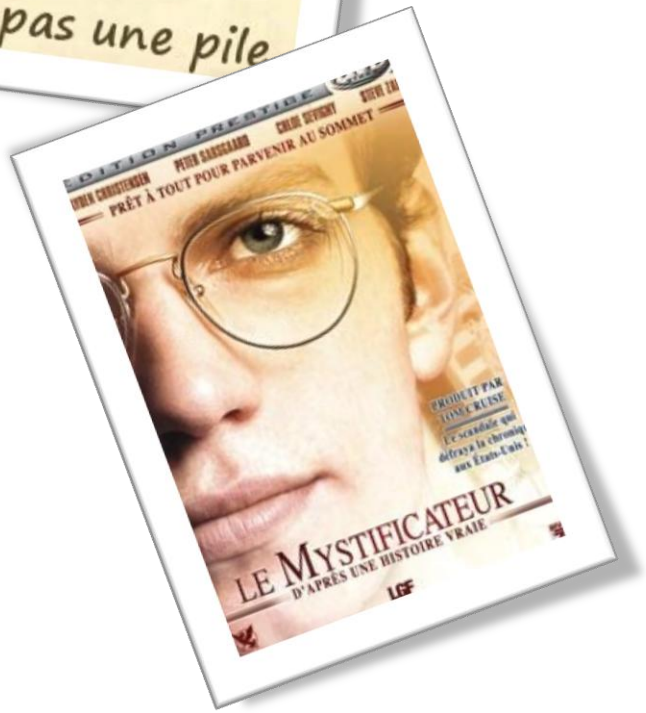
Contexte

Théorème :

« Vivons heureux, vivons caché »

Corolaire :

« Si une machine peut falsifier son identité au yeux des outils de reconnaissance et usurper celle d'un système moins intéressant pour la menace, celle ci minimise l'attrait de l'attaquant et perturbe la pertinence des attaques ciblées à sa nature apparente.»



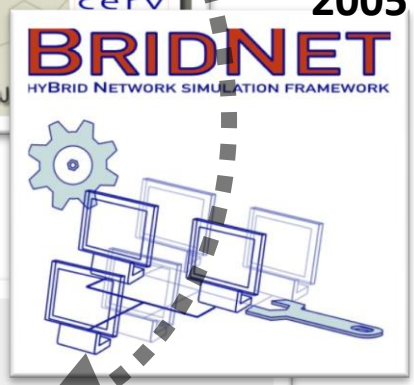
IpMorph : « vers la mystification de la prise d'empreinte de pile »

Historique

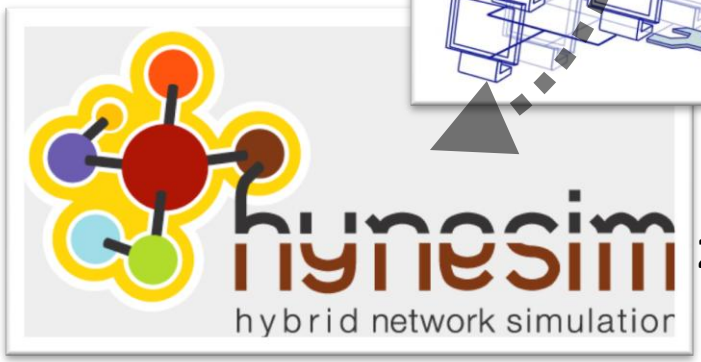
- **GFOX-0 & GFOX-1**
  - Simulation multi-agents de la confidentialité, de l'intégrité et de la disponibilité d'un système d'information
- **EREBOR**
  - Vers l'hybride « réel/virtuel »
- **BridNet**
  - Première pile « UserLand TCP/IP »
- **Hynesim**
  - Plateforme de virtualisation de système d'information

1999

2002



2005



2008-?

IpMorph : « vers la mystification de la prise d'empreinte de pile »

Etymologie

- IpMorph
  - Du néo-grec numérique μορφή - θημωνιά, *morphéstakis*, « forme de pile ».
- Suffixe
  - -morph /mɔʁf/

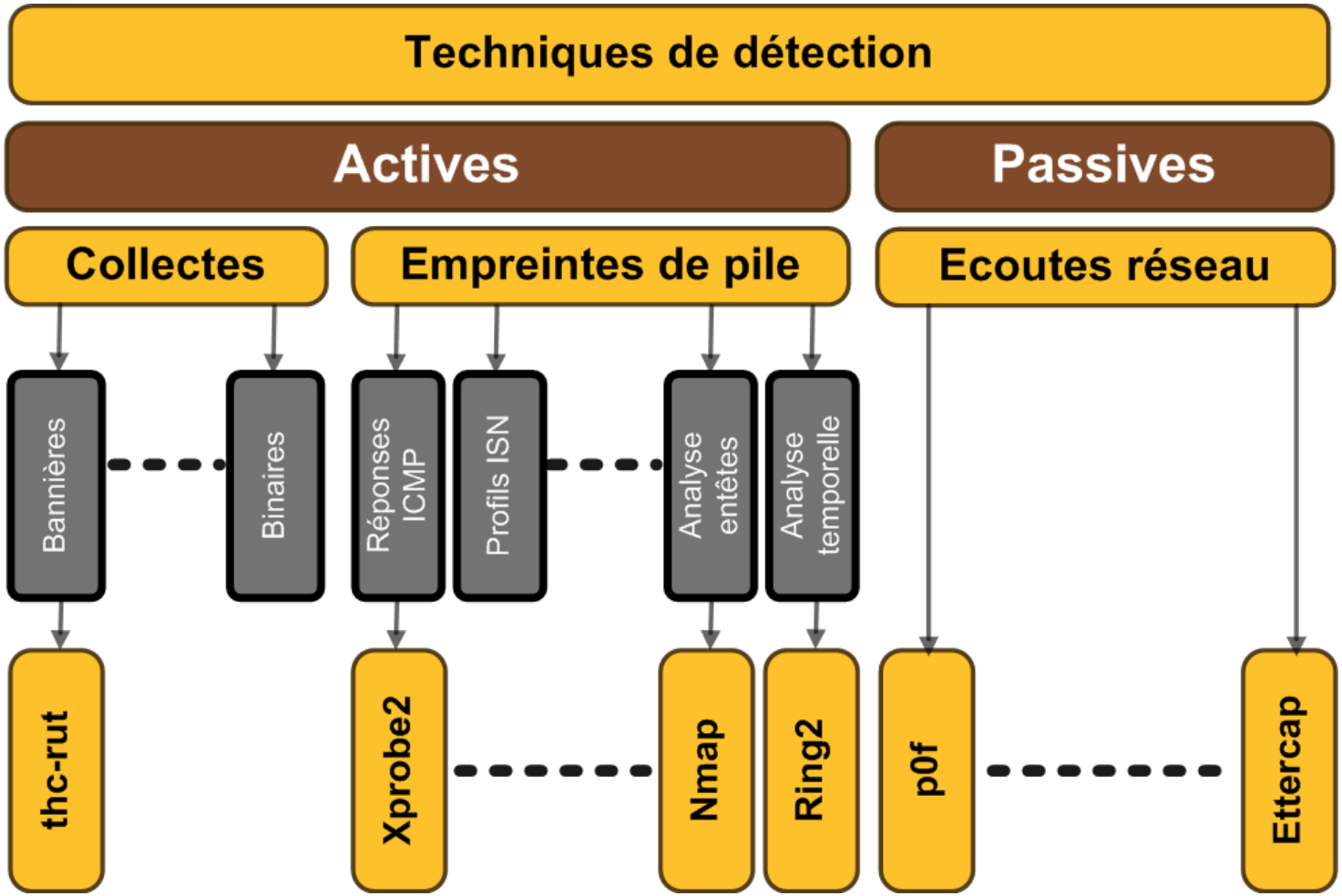
1. En relation avec la forme d'une pile IP, qui a la forme d'une pile IP.

- Apparentés étymologiques

1. FpFucker
2. OS fingerprint frustrating

IpMorph : « vers la mystification de la prise d'empreinte de pile »

Techniques de détection *[11]*

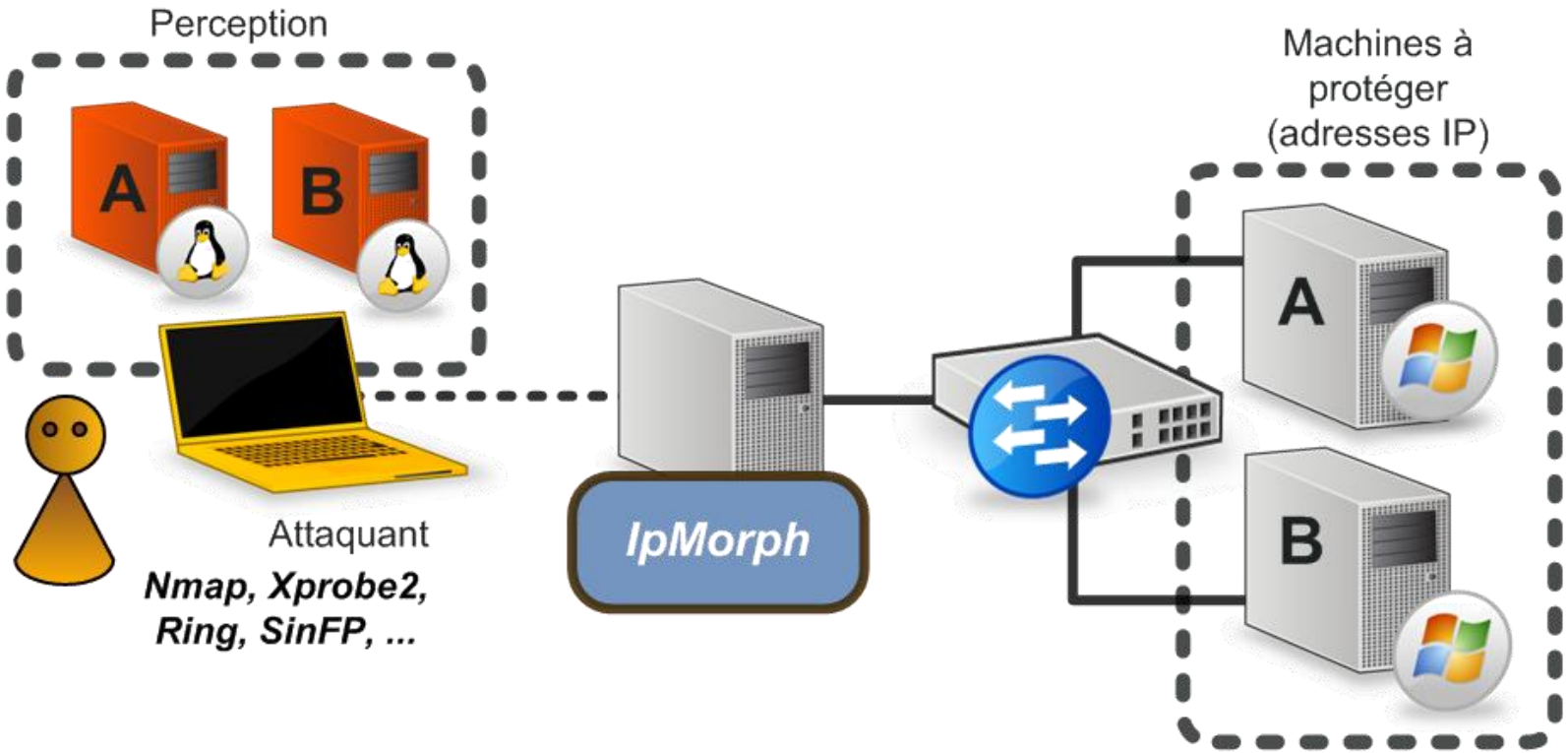


Etat de l'art de la mystification [7]

- **Filtrage**
  - Stealth patch : **Unmaintained as of 2002, GNU/Linux kernel 2.2-2.4** *[14]*
  - Blackhole : **FreeBSD, kernel options** *[16]*
  - IPlog : **Unmaintaned as of 2001, *BSD** *[17]*
  - Packet filter : **OpenBSD** *[18]*
- **Configuration et modification de pile TCP/IP ("host based")**
  - Ip Personality *[19]*
  - Fingerprint Fucker *[12][13]*
  - Fingerprint scrubber *[1]*
  - OSfuscate *[8]*
- **Substitution de pile TCP/IP ("proxy behaviour")**
  - Honeyd *[9]*
  - Packet purgatory / Morph *[10]*

IpMorph : « vers la mystification de la prise d'empreinte de pile »

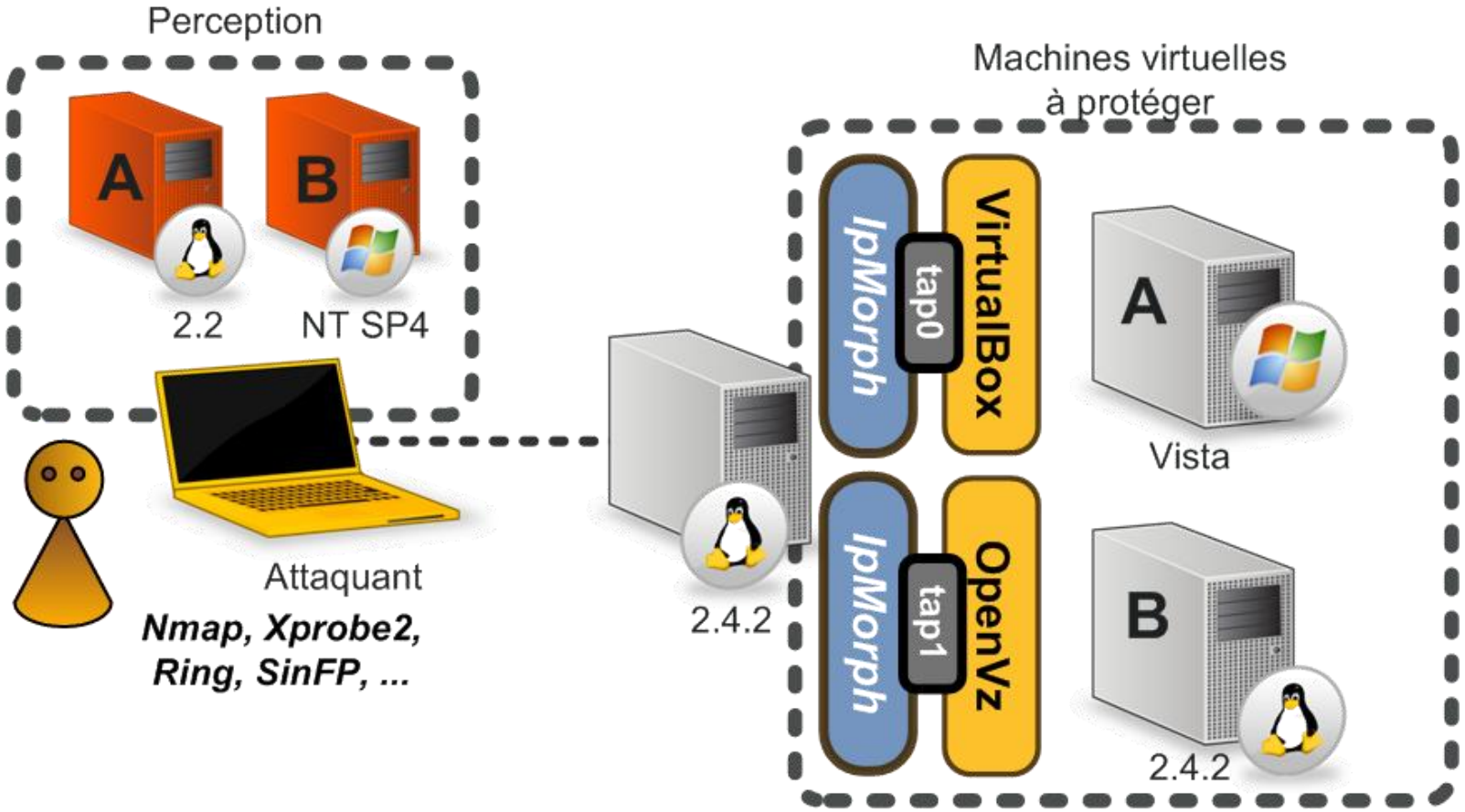
Utilisation d'IpMorph 1/2



IpMorph en coupure de machines réelles

IpMorph : « vers la mystification de la prise d'empreinte de pile »

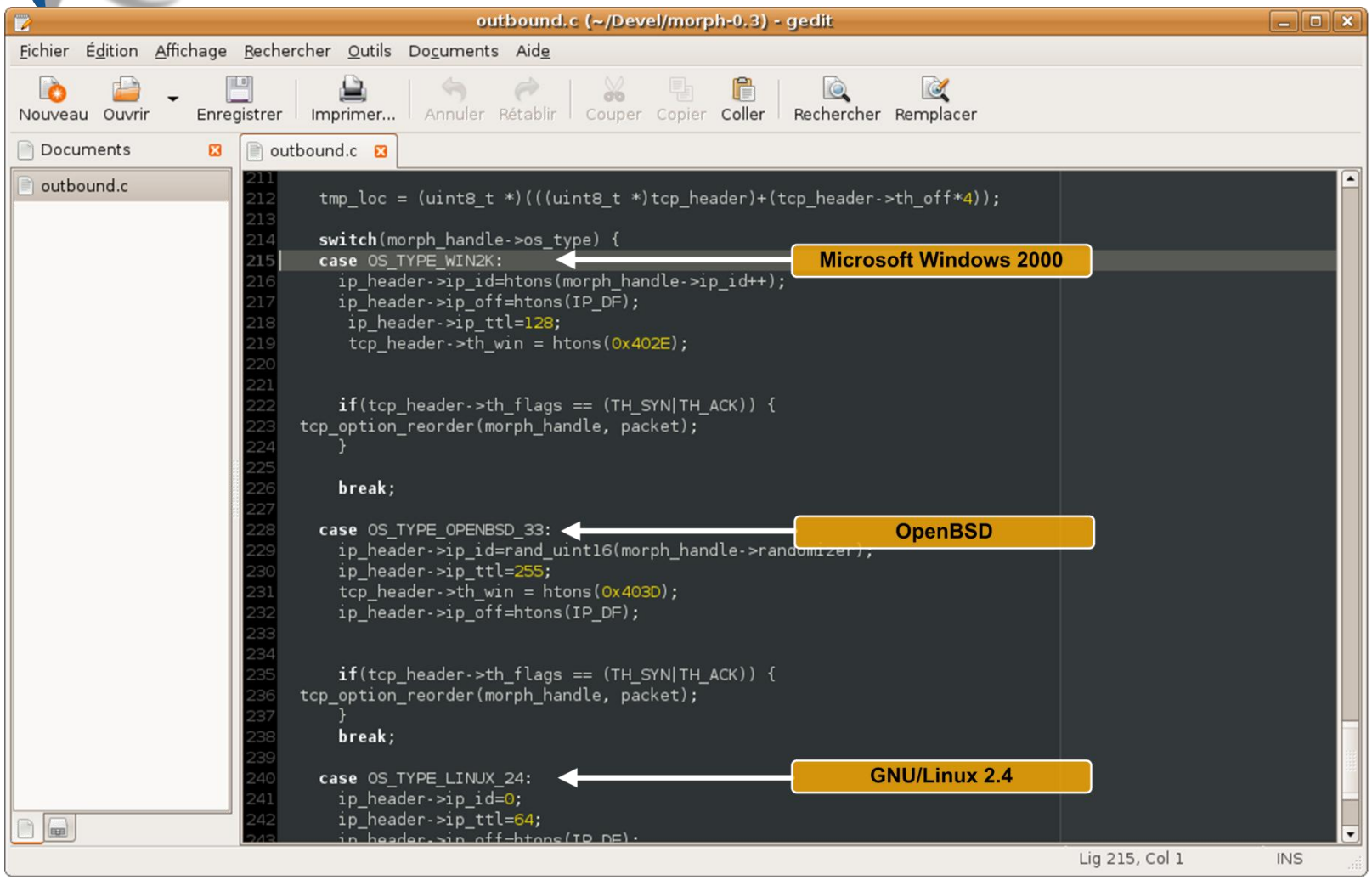
Utilisation d'IpMorph 2/2



IpMorph en relai protection de machines virtuelles

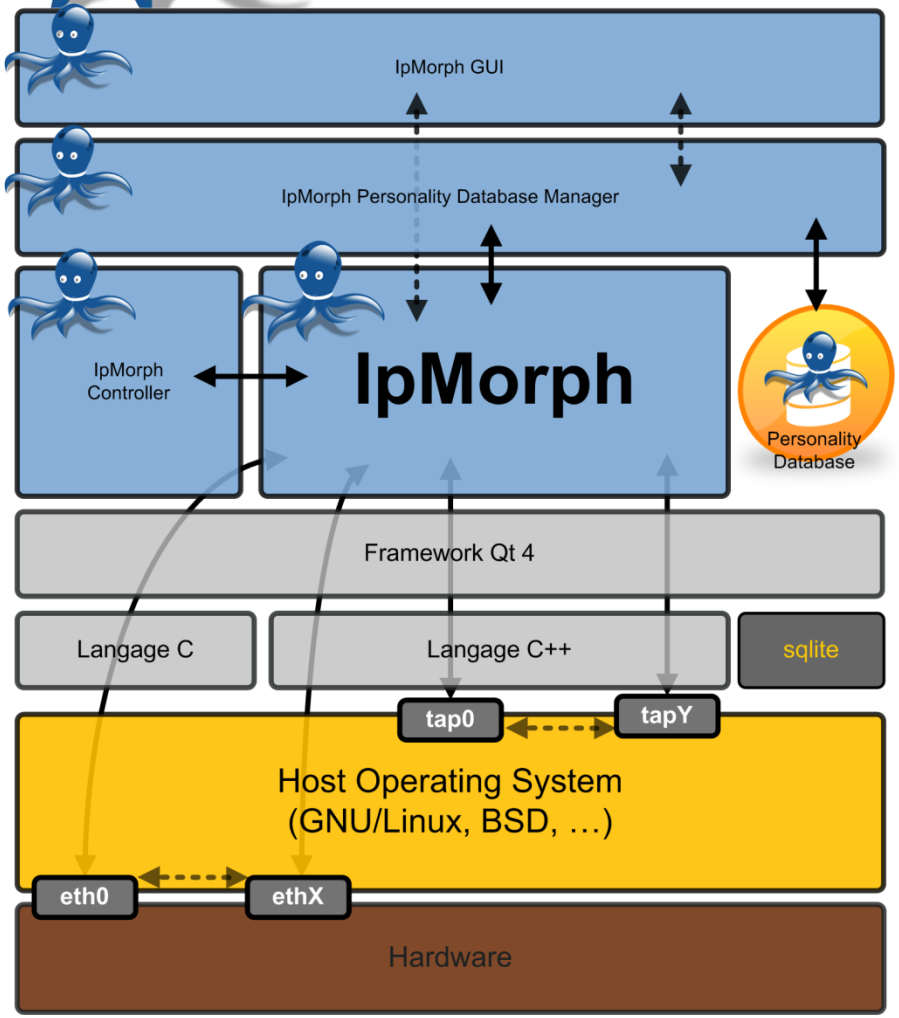
IpMorph : « vers la mystification de la prise d'empreinte de pile »

IpMorph != Packet Purgatory / Morph *[20]*



IpMorph : « vers la mystification de la prise d'empreinte de pile »

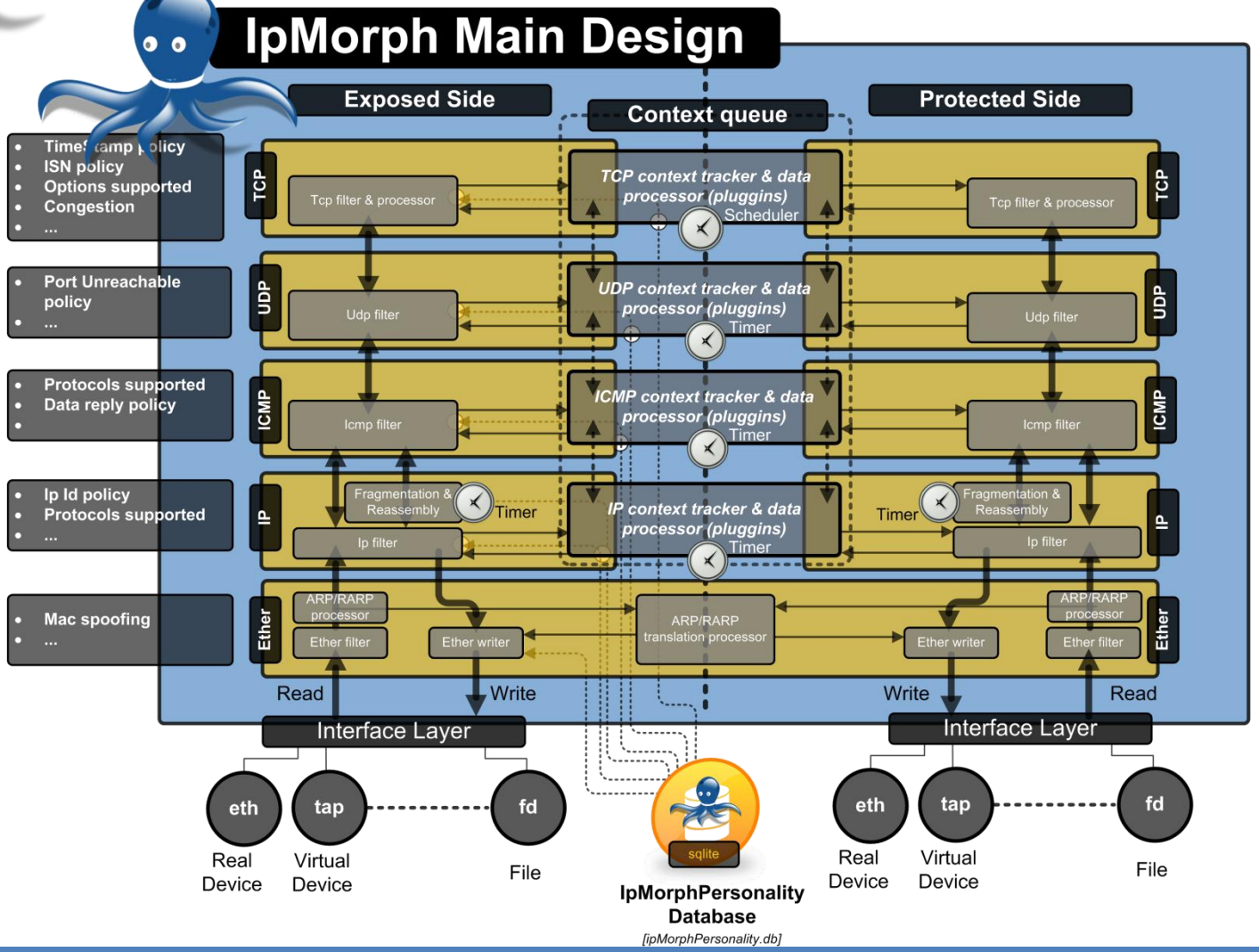
Socle logiciel



- **Langage C++**
- **Application « UserLand »**
- **Framework Qt4**
- **Eléments constituants :**
  - IpMorph (Core)
  - IpMorph Controller
  - IpMorph Personality Manager
  - IpMorph Personality Database
  - IpView (IpMorph GUI)
- **Portabilité :**
  - GNU/Linux
  - *BSD, Mac OS
- **License GPLv3**

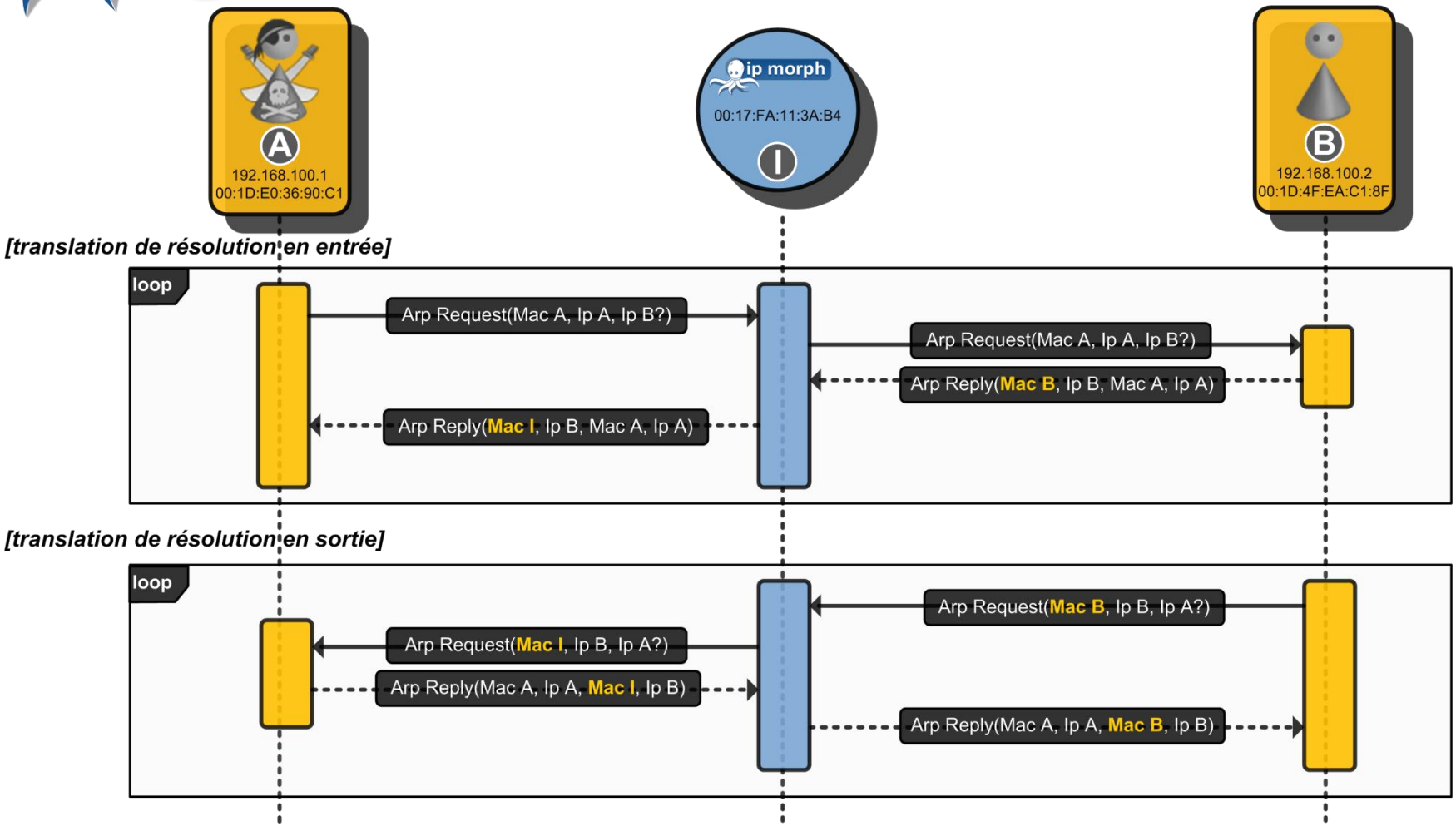
IpMorph : « vers la mystification de la prise d'empreinte de pile »

Architecture générale



IpMorph : « vers la mystification de la prise d'empreinte de pile »

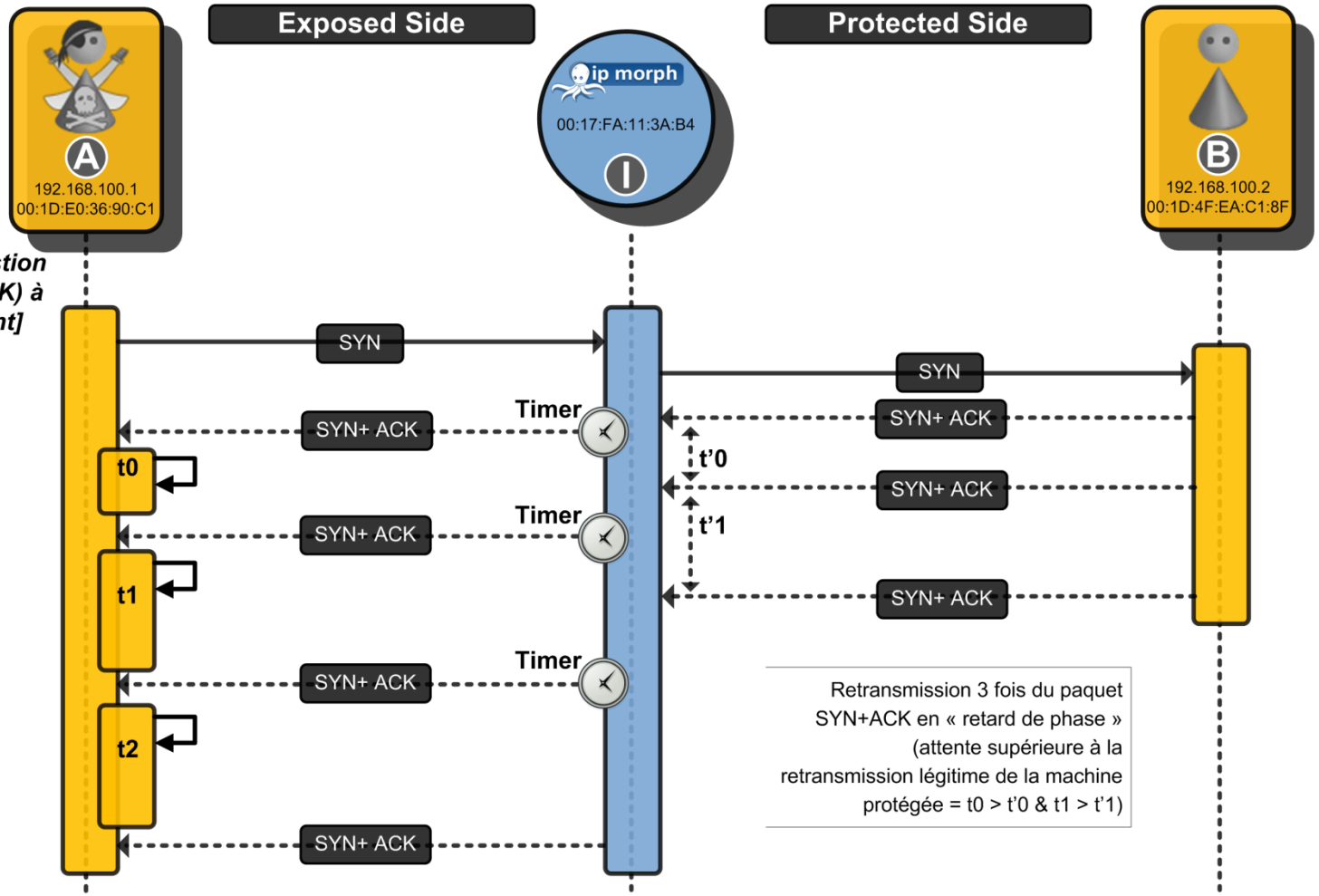
Translation ARP



IpMorph : « vers la mystification de la prise d'empreinte de pile »

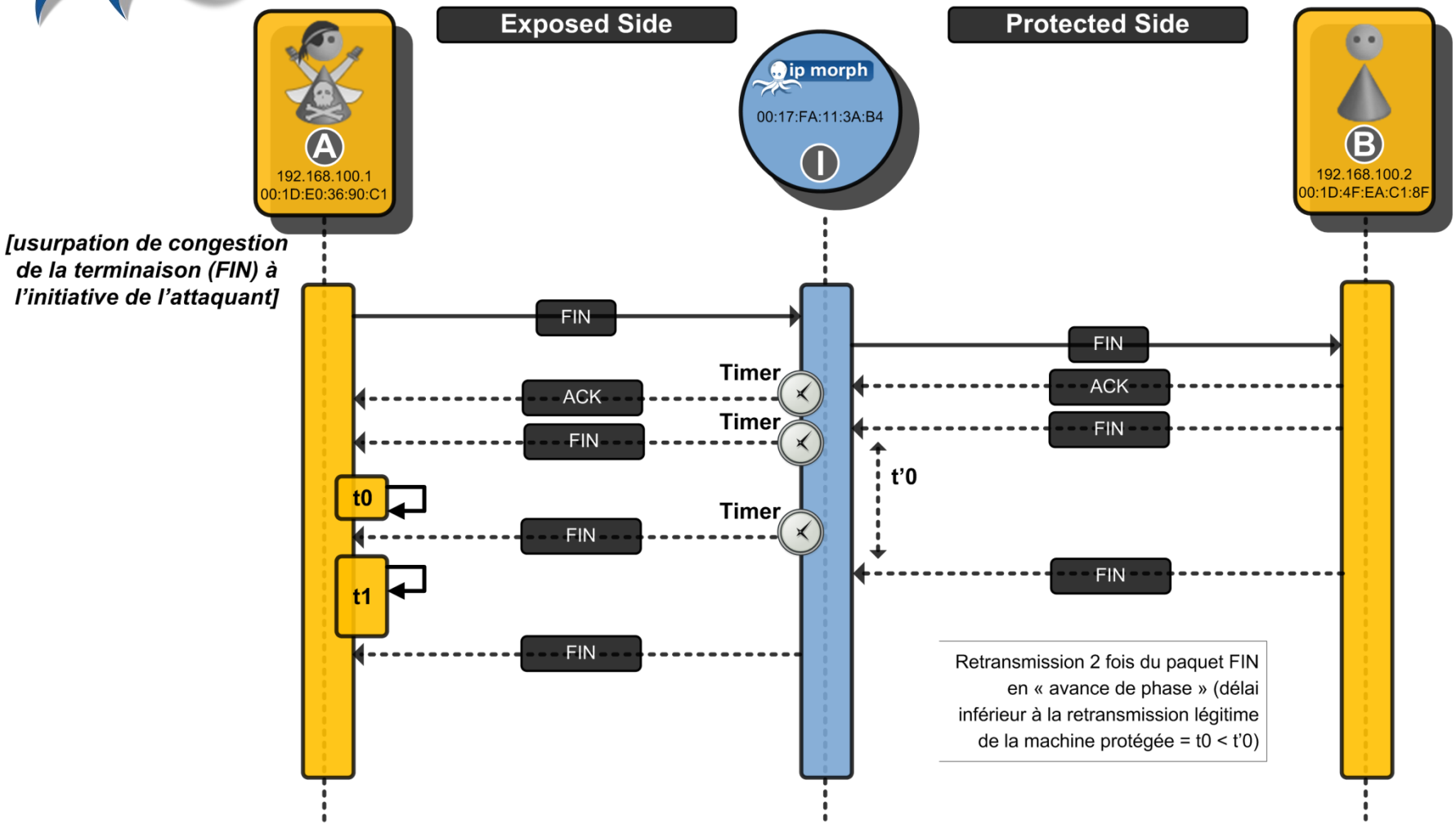
Mystification de la congestion - SYN+ACK

[usurpation de la congestion sur connexion (SYN+ACK) à l'initiative de l'attaquant]



IpMorph : « vers la mystification de la prise d'empreinte de pile »

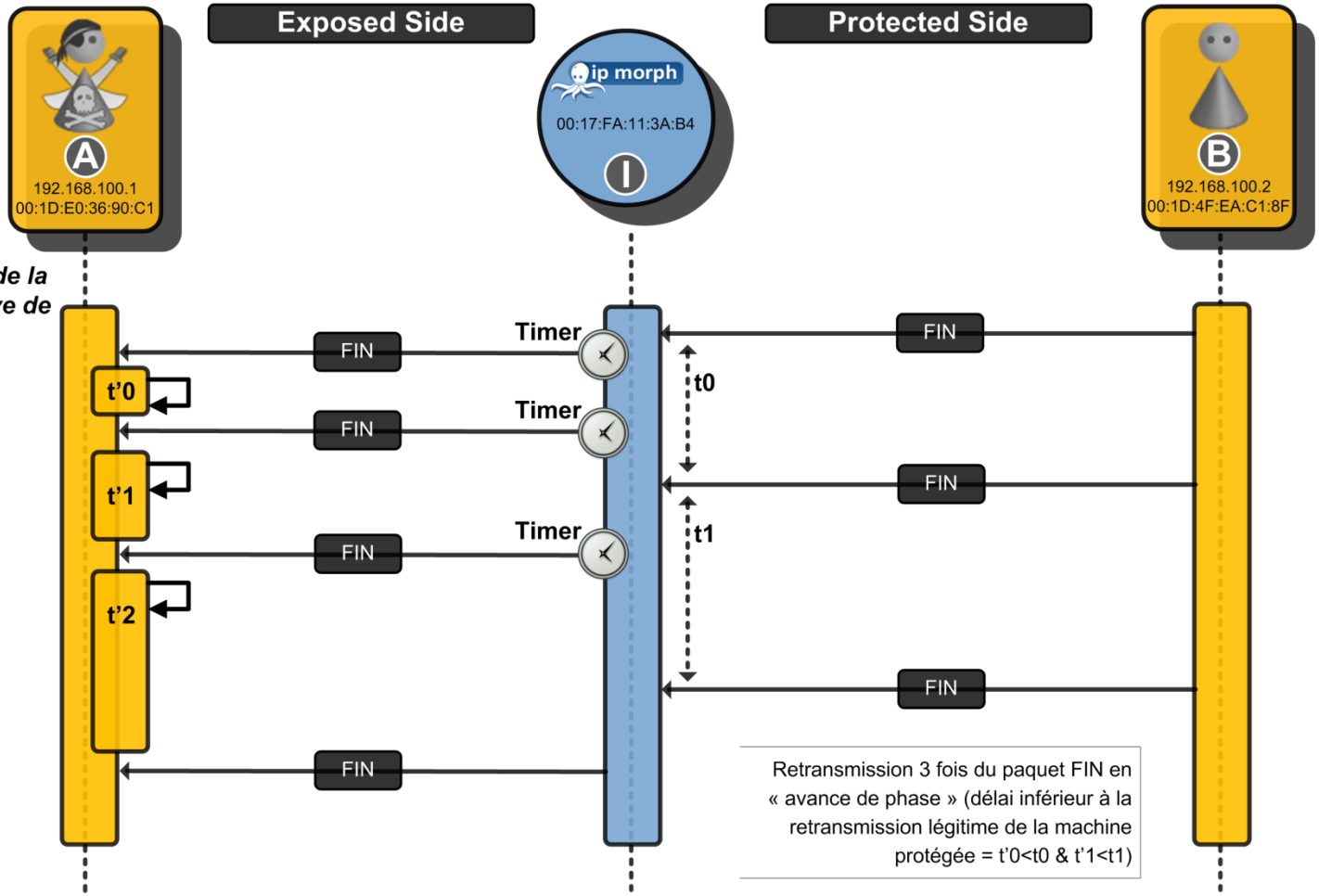
Mystification de la congestion - FIN+ACK « client »



IpMorph : « vers la mystification de la prise d'empreinte de pile »

Mystification de la congestion - FIN+ACK « serveur »

[usurpation de congestion de la terminaison (FIN) à l'initiative de la machine protégée]



IpMorph : « vers la mystification de la prise d'empreinte de pile »

Focus : Calcul ISN

ipMorphTcpContext.cpp (~) - gedit

Fichier Édition Affichage Rechercher Outils Documents Aide

Nouveau Ouvrir Enregistrer Imprimer... Annuler Rétablir Couper Copier Coller Rechercher Remplacer

ipMorphTcpContext.cpp

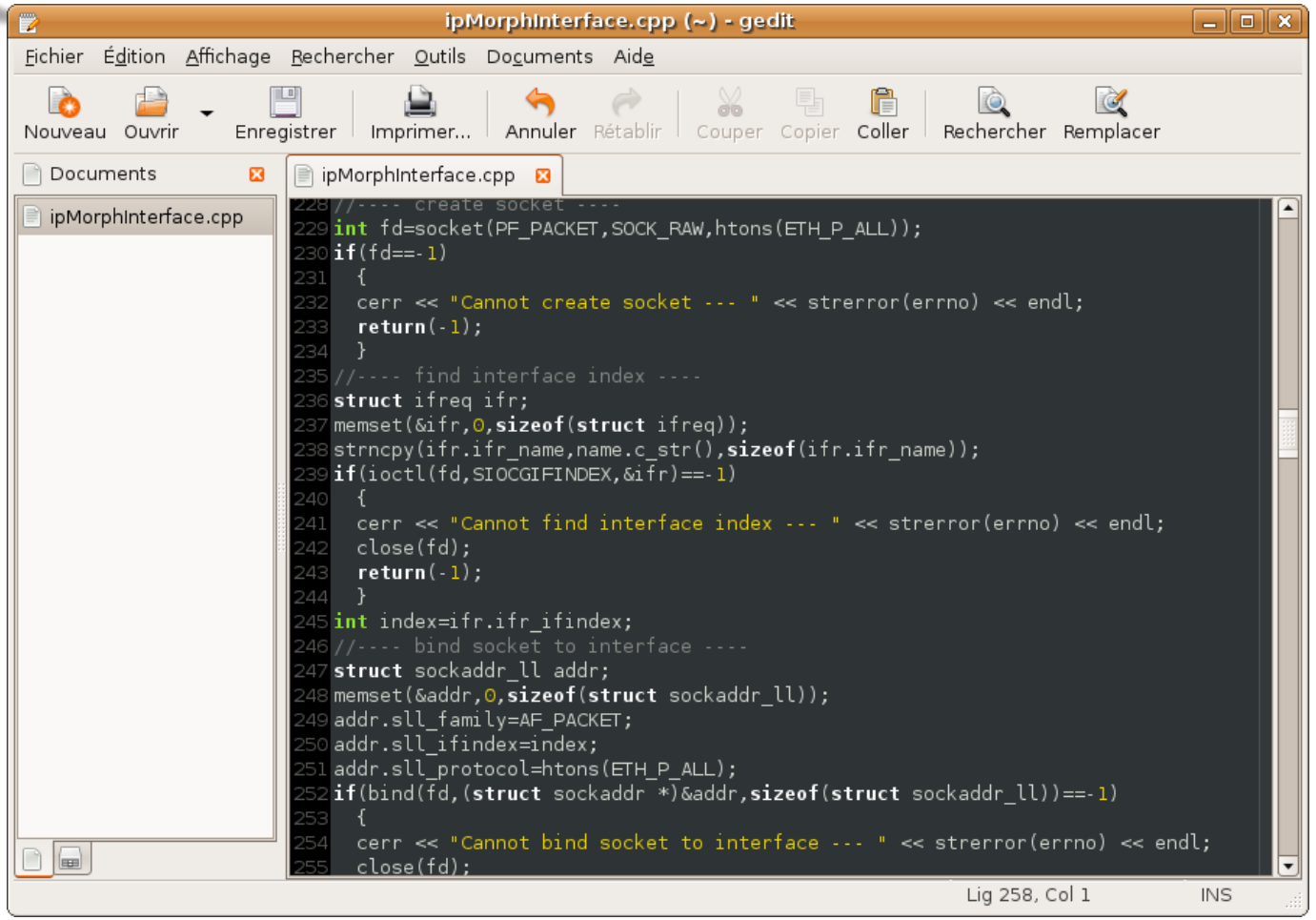
```
1209  if(_NOS.getTcpIsnMode() == NmapOS::ISN_TR) // True Random
1210  {
1211    _isn = rand();
1212  }
1213  else if(_NOS.getTcpIsnMode() == NmapOS::ISN_RI) // Random Increment
1214  {
1215  // méthode de Box-Muller
1216  // r et phi aléatoires entre 0 et 1 : (0 1]
1217  // calcul des nombres de la loi normale :
1218  //    z0 = cos(2*PI*phi) * sqrt(-2*ln(r))
1219  //    z1 = sin(2*PI*phi) * sqrt(-2*ln(r))
1220  // et on obtient nos 2 nombres avec le bon écart-type autour de la moyenne :
1221  //    n0 = moyenne + ecart-type * z0
1222  //    n1 = moyenne + ecart-type * z1
1223  // note : en C, ln() correspond à la fonction "log()" de math.h
1224
1225    unsigned long ecartType = _NOS.getTcpIsnParm();
1226    unsigned long moyenne = 2*ecartType;
1227
1228    srand(_getTime() + _isn); // réinitialisation de la génération des nombres aléatoires
1229    double r   = ((double)1*(1+rand())/(RAND_MAX+1.0));
1230
1231    srand(_getTime() + (int)(_isn * (1+r))); // réinitialisation de la génération des nombres aléatoires
1232    double phi = ((double)1*(1+rand())/(RAND_MAX+1.0));
1233
1234    double z0 = cos(2*M_PI*phi) * sqrt(-2*log(r));
1235    double n0 = moyenne + ecartType * z0;
1236
1237    _isn = (long)n0;
1238  }
1239  else if(_NOS.getTcpIsnMode() == NmapOS::ISN_TD) // Time Dependant
1240  {
1241    unsigned int gcd = _NOS.getTcpIsnParm();
1242    if(gcd!=0)
1243    {
1244      _isn+=gcd;
1245    }
1246    else
1247    {
1248      unsigned int avg = 10;
1249      _isn+=avg+lrint((double)avg*rand()/(RAND_MAX+1.0));
1250    }
1251  }
1252  else if(_NOS.getTcpIsnMode() == NmapOS::ISN_64K) // 64K
1253  {
1254    _isn+=64000;
1255  }
1256  else if( NOS.getTcpIsnMode() == NmapOS::ISN_i800) // i800
```

Lig 1235, Col 1 INS

Génération de l'ISN pour l'ancienne version de Nmap (<4.11)

IpMorph : « vers la mystification de la prise d'empreinte de pile »

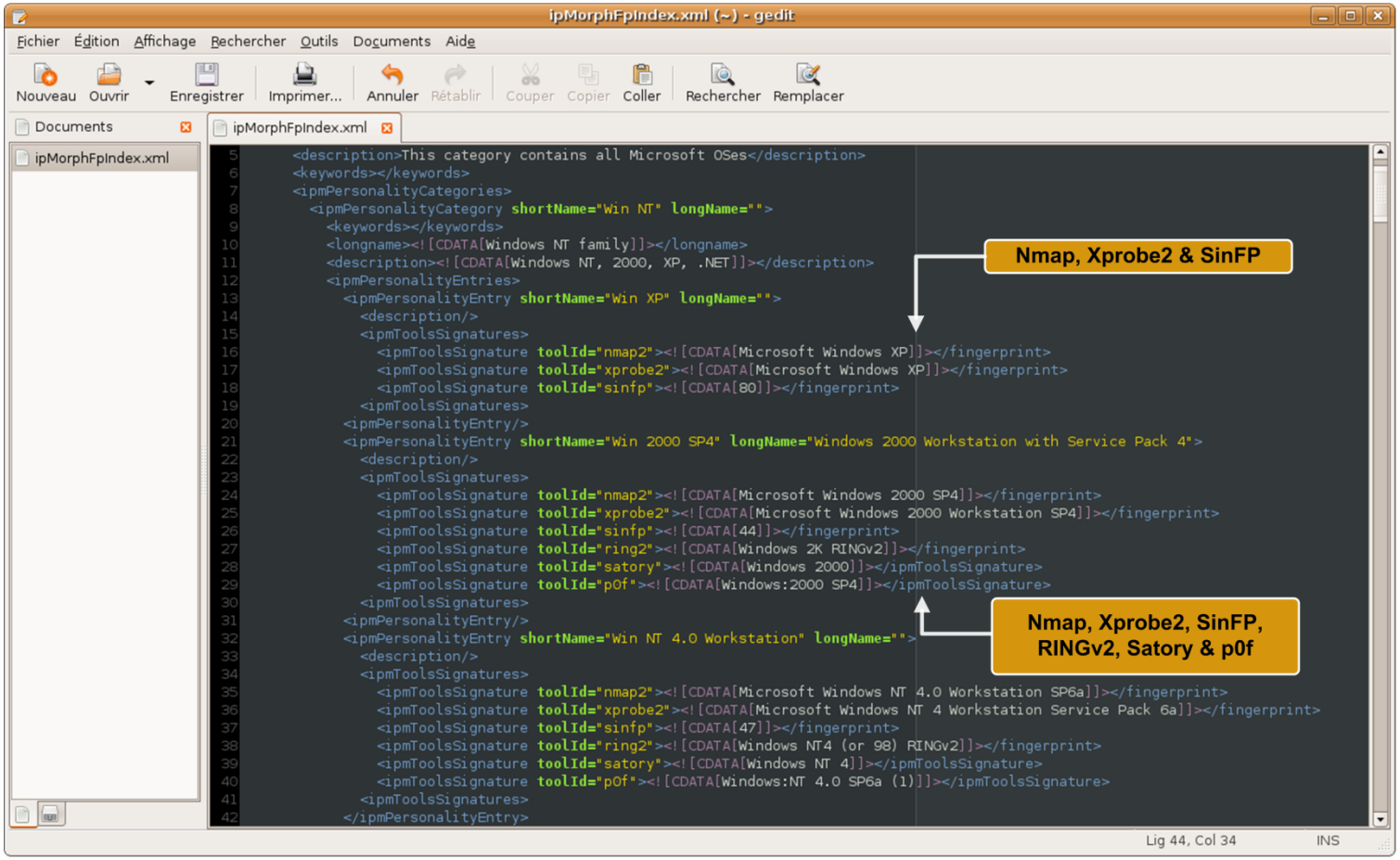
Focus : Interfaces réseaux « read/write »



Ouvertures des interfaces (eth, tap) pour capture et injection

IpMorph : « vers la mystification de la prise d'empreinte de pile »

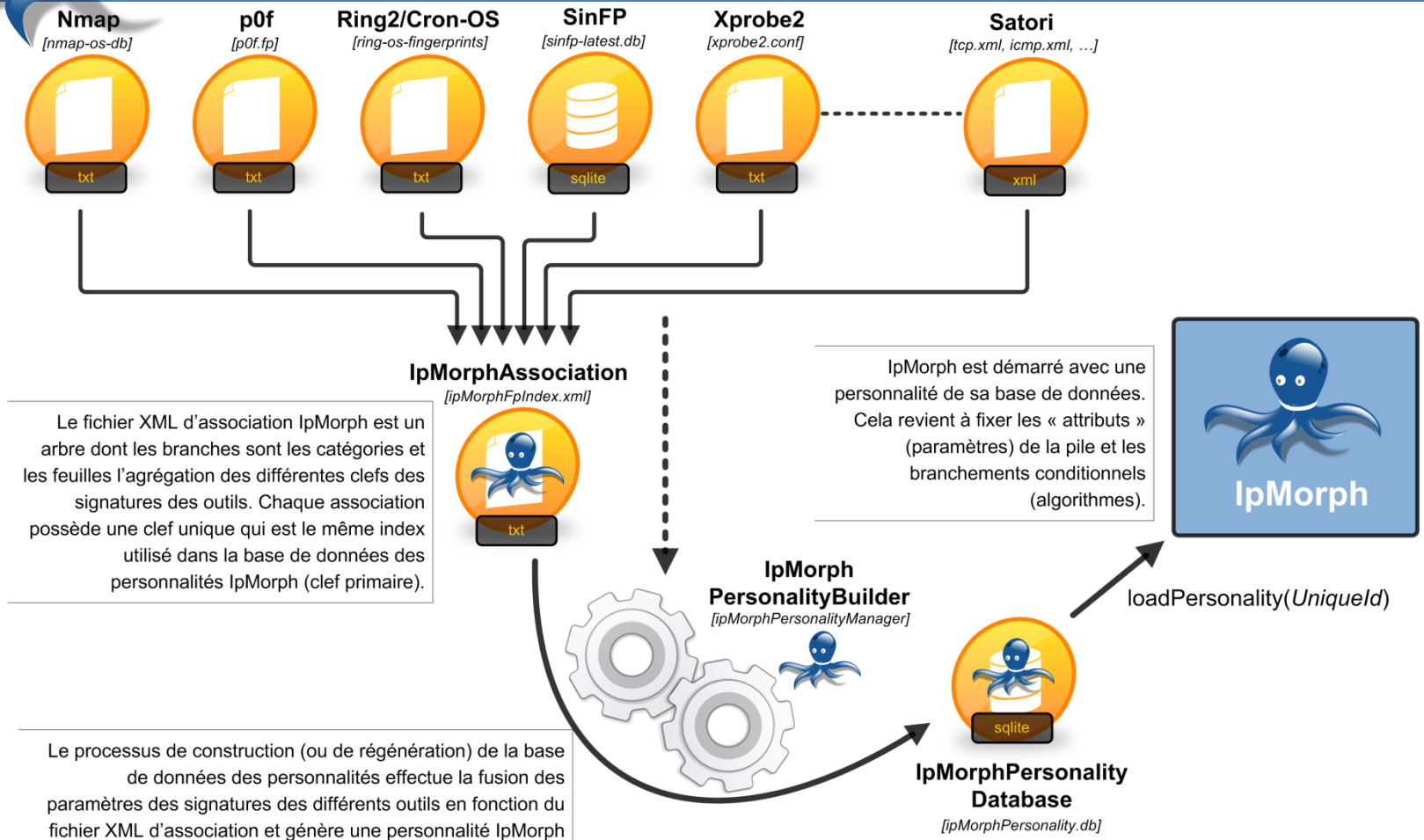
Personality Manager : Structure de l'index



Formalisation XML et association des signatures des outils d'OSFP

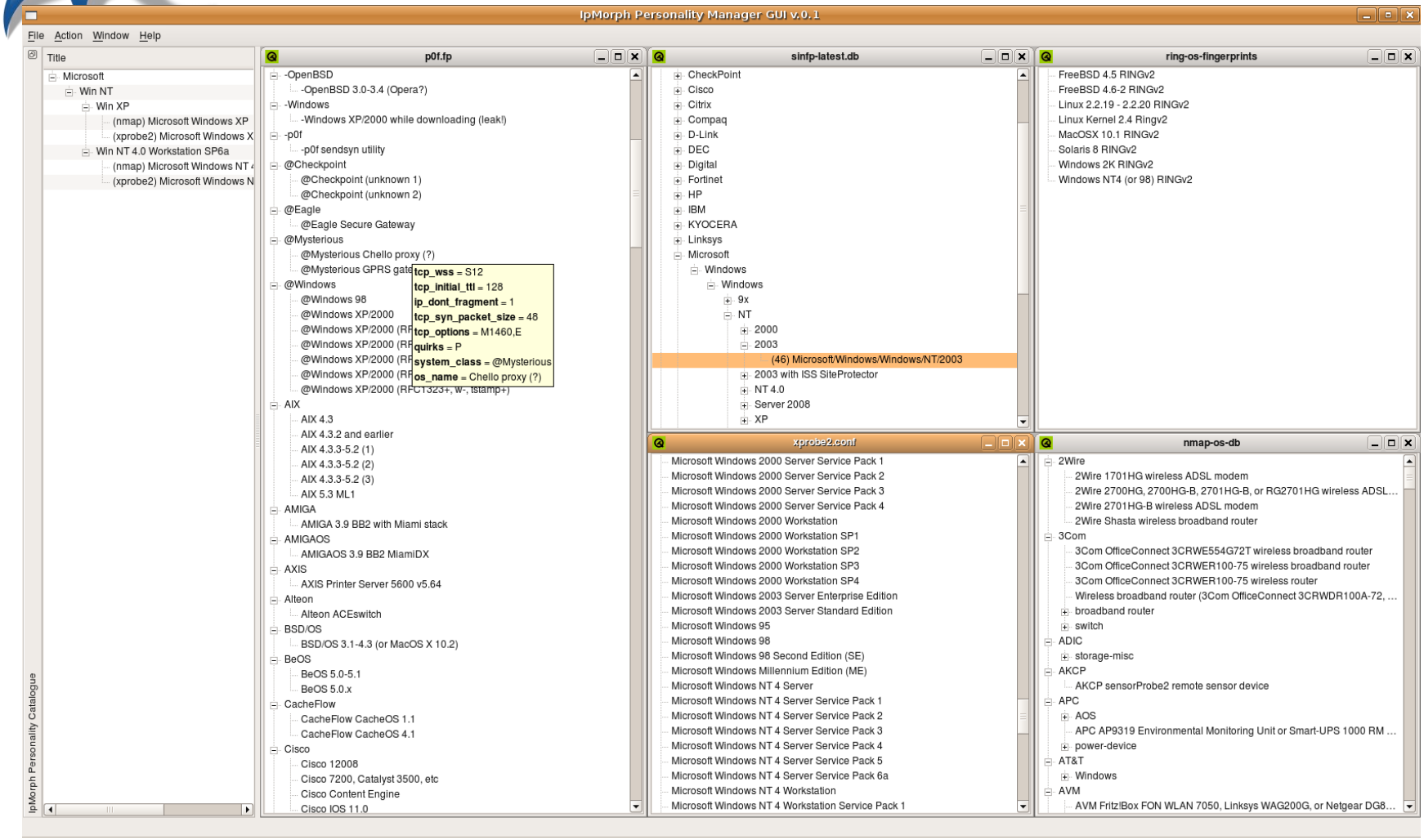
IpMorph : « vers la mystification de la prise d'empreinte de pile »

Personality Manager : Présentation



IpMorph : « vers la mystification de la prise d'empreinte de pile »

Personality Manager : Démonstration



IpMorph : « vers la mystification de la prise d'empreinte de pile »

Personality Manager : Database Overview

XProbe2Signature

```
icmp_echo_reply_code: bool
icmp_echo_reply_ip_id: int
icmp_echo_reply_tos_bits: bool
icmp_echo_reply_df_bit: bool
icmp_echo_reply_ttl: int

icmp_timestamp_reply_received: bool
icmp_timestamp_reply_ttl: int
icmp_timestamp_reply_ip_id: int

icmp_address_mask_reply_received: bool
icmp_address_mask_reply_ttl: int
icmp_address_mask_reply_ip_id: int

icmp_information_reply_received: bool
icmp_information_reply_ttl: int
icmp_information_reply_ip_id: int

icmp_udp_unreach_reply_echoed_dtsize: int
icmp_udp_unreach_reply_ttl: int
icmp_udp_unreach_reply_precedence_bits: int
icmp_udp_unreach_reply_df_bit: bool
icmp_udp_unreach_reply_ip_id: int
icmp_udp_unreach_reply_payload_udp_chksum: int
icmp_udp_unreach_reply_payload_ip_chksum
icmp_udp_unreach_reply_payload_ip_df: bool
icmp_udp_unreach_reply_payload_total_len: int
icmp_udp_unreach_reply_payload_3bit_flags: bool

tcp_syn/ack_reply_tos: int
tcp_syn/ack_reply_df: bool
tcp_syn/ack_reply_ip_id: int
tcp_syn/ack_reply_ttl: int
tcp_syn/ack_reply_ack_delta: int
tcp_syn/ack_reply_wss: int
tcp_syn/ack_reply_options: byte[]
tcp_syn/ack_reply_wscale: int
```

Nmap2Signature

```
tcp_t1_ecn_reply: int[]
tcp_t2_reply: int[]
tcp_t3_reply: int[]
tcp_t4_reply: int[]
tcp_t5_reply: int[]
tcp_t6_reply: int[]
tcp_t7_reply: int[]
icmp_ie_reply: int[]
udp_u1_reply: int[]
```

SinFPSignature

```
ip_ttl: bool
ip_id: int
ip_df: bool
tcp_seq: int
tcp_ack: int
tcp_flags: byte[]
tcp_wss: int
tcp_mss: int
tcp_options: byte[]
```

p0fSignature

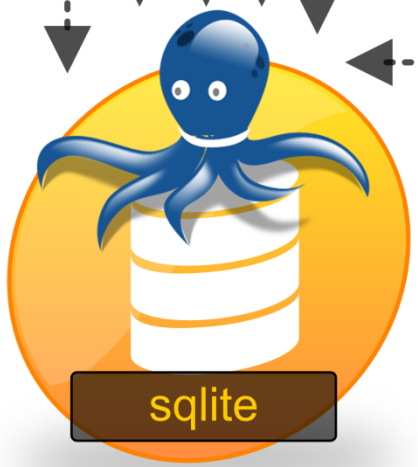
```
ip_dont_fragment: bool
tcp_wss: int
tcp_packet_size: int
tcp_initial_ttl: int
tcp_mss: int
tcp_window_scaling: int
tcp_options: byte[]
tcp_quirks: bool[]
```

Ring2Signature

```
nb_packets: int
ring_timeout: int
connexion_time: int
retry_intervals: int[]
```

myToolSignature

?



IpMorphPersonalityDatabase

[ipMorphPersonality.db]

IpMorph : Illustration pratique

DEMONSTRATION

IpMorph : « vers la mystification de la prise d'empreinte de pile »

Etat d'avancement

- **Conception = 70%**
  - Etat de l'art bibliographique = 85%
  - Etat de l'art technique mystification = 70%
  - Analyse des outils d'OSFP = 70%
    - ***Nmap = 80%***
    - ***Xprobe2 = 95%***
    - ***SinFP = 60%***
    - ***Satori = 45%***
    - ***Ring2 = 80%***
  - Architecture générale = 80%
  - Spécification & UML/Software design= 50%
- **Implémentation = 10%**
  - Proof of concept « trash programming » = 85%
  - Développement des composants = 5% (50% pour Personality Manager)
  - *Tests unitaires, tests fonctionnels, documentation = 5%*

IpMorph : « vers la mystification de la prise d'empreinte de pile »

Perspectives

- **Juin 2009 – SSTIC 2009**
  - Beta release 0.5 en download
  - Site web, Mailing list, Bugtracking
  - Subversion / sources
- **Décembre 2009**
  - Version 1.0 en download
  - Documentation, UserGuide, DevGuide
  - Distribution .deb, .rpm, …
  - Intégration de quelques scrubbers applicatifs (DNS, SMB, DHCP, …).

IpMorph : « vers la mystification de la prise d'empreinte de pile »

Bibliographie 1/2

1. ***Smart, M., Malan, G.R., Jahanian, F.***: **Defeating TCP/IP Stack Fingerprinting**, 9th USENIX Security Symp.
2. ***Smith, C., Grundl, P.*** : **Know Your Enemy: Passive Fingerprinting**, http://old.honeynet.org/papers/finger/, (2002)
3. ***Fyodor*** : **Remote OS detection via TCP/IP Stack FingerPrinting**, Phrack 1998, http://www.insecure.org/nmap/nmap-fingerprinting-article.txt, (1998)
4. ***Spangler, R.***: **Analysis of Remote Active Operating System Fingerprinting Tools**, http://www.packetwatch.net/documents/papers/osdetection.pdf
5. ***Veysset, F., Courtay, O., Heen, O.***: **New Tool And Technique For Remote Operating System Fingerprinting**, http://www.hackerz.ir/e-books/remote_os_detection.pdf, (2002)
6. ***Auffret, P.*** : ***SinFP, Unification de la prise d'empreinte passive et active des systèmes d'exploitation***, http://actes.sstic.org/SSTIC08/SinFP_Unification_Prise_Empreinte_Active_Passive_Systemes_Exploitation/ , SSTIC 08, (2008)
7. ***Berrueta, D.B.***: **A practical approach for defeating Nmap OS-Fingerprinting**, http://nmap.org/misc/defeat-nmap-osdetect.html (2003)
8. ***Crenshaw, A.*** : **OSfuscate: Change your Windows OS TCP/IP Fingerprint to confuse P0f, NetworkMiner, Ettercap, Nmap and other OS detection tools**, http://www.irongeek.com/i.php?page=security/osfuscate-change-your-windows-os-tcp-ip-fingerprint-to-confuse-p0f-networkminer-ettercap-nmap-and-other-os-detection-tools, (2008)
9. ***Provos, N.*** : **Honeyd: A Virtual Honeypot Daemon**, http://www.citi.umich.edu/u/provos/papers/honeyd-eabstract.pdf , (2003)
10. ***Wang, K.*** : **Frustrating OS Fingerprinting with Morph**, http://www.synacklabs.net/projects/morph/Wang-Morph-TheFifthHOPE.pdf, The Fifth HOPE, (2004)
11. ***Veysset, F., Courtay, O., Heen, O.***: **Détection des systems d'exploitation avec RINGv2**, Actes SSTIC, (2003)
12. ***Fusys,CyRaX*** : ***Fingerprint Fucker***, http://www.s0ftpj.org/tools/fingfuck.tgz
13. ***Reed, D.*** : **Fingerprint Fucker**, http://packetstormsecurity.org/UNIX/misc/bsdfpf.tar.gz
14. ***Trifero, S., Callaway, D.*** : **Linux Kernel Stealth Patch**, http://www.innu.org/~sean/

IpMorph : « vers la mystification de la prise d'empreinte de pile »

Bibliographie 2/2

15. ***Aubert, S.*** : **Antimap**, http://www.hsc.fr/ressources/breves/antimap.html
16. ***FreeBSD***, **Blackhole**, http://www.gsp.com/cgi-bin/man.cgi?section=4&topic=blackhole
17. ***McCabe, R.*** : **Iplog**, http://ojnk.sourceforge.net/stuff/iplog.readme
18. ***OpenBSD*** : **PF « le filtrage de paquet »**, http://www.openbsd.org/faq/pf/fr/filter.html
19. **Roualland, G., Saffroy, J-M.** : **IP Personality**, http://ippersonality.sourceforge.net/
20. **Wang, K.** : **Frustrating OS Fingerprinting with Morph**, DEFCON 12, http://www.synacklabs.net/projects/morph/Wang-Morph-DEFCON12.pdf (2004)

Merci de votre attention.

Interrogations ? Critiques !

Suggestions…

Nous contacter :

contact@ipmorph.org

Beta Release 2009-06

http://dev.ipmorph.org