



services de **confiance**  
et **sécurité**  
Dématérialisation  
**Management** sécurité  
vulnérabilités et attaques  
**Ethique** et sécurité  
Normes et standards  
Cybercriminalité  
Veille stratégique



# Honeypot Farms

# Honeyynet

EUROSEC 2004

# Project Farm

Cédric Blancher / Franck Veysset

# Plan



- S Le concept de Honeypot
  - Q Qu'est-ce qu'un honeypot ?
  - Q Principes et objectifs
- S Quelques contraintes liées aux HP
  - Q Déploiement et administration
  - Q Couverture, ressources
- S Une solution : Les « honeypot farms »
  - Q Définition
  - Q Principes et usages
- S Exemple de réalisation
  - Q Prototype d'une solution
  - Q Détails techniques

# Honeypot ?



## S Principe

- Q Mettre en place un système « leurre » destiné à étudier les activités anormales / hostiles survenant sur un réseau (contre ce système)
- Q Observer / surveiller ce système
- Q Pour apprendre les tactiques / motivations / outils utilisés par les « hackers » sur l'Internet (*peut aussi se décliner en Intranet*)

## S Thème d'actualité : beaucoup d'articles sur le sujet

- Q Nouvel axe prometteur de la communauté « sécurité »
- Q Des articles dans la presse généraliste
  - Comment les honeypots leurrent les pirates, 3 septembre 01 sur 01NET
- Q Le « French Honeynet Project »

# Principes



**S** Il faut voir le honeypot comme un « microscope »

**Q** Il ne voit pas tout

**Q** Mais permet une analyse très fine sur les éléments collectés

... qui sont par nature plutôt suspects



# Objectifs (1/2)



**S** Avant tout, apprendre

**Q** Que se passe-t-il vraiment sur les réseaux

**Q** Que font les hackers ?

**Q** Que cherchent-ils ?

**Q** Quels sont leurs armes ?

**Q** L'espionnage industriel : mythe ou réalité ?

**Q** ...

**S** Apprendre à mieux détecter



**S** Pour mieux se protéger



# Objectifs (2/2)



- S** Pour la recherche
  - Q** Connaître les tendances dans le domaine des attaques
  - Q** Connaître ses ennemis
  - Q** Capturer de nouveaux outils (worm...)
- S** Pour sécuriser son environnement
  - Q** Détection des nouvelles attaques
- S** Pour se préparer en cas d'attaques sur les réseaux opérationnels
- S** Et pour apprendre à se défendre



# Problématiques (1/2)



## S Utilisation des honeypots

Q Nécessité de maximiser les chances de capturer des événements intéressants

Q Choix de l'emplacement du honeypot dans le réseau

- En interne ?
- Devant le firewall ?
- En DMZ ?

Q Pas de solutions, dépend des objectifs...

S Nécessité de déployer plusieurs honeypots, et de « surveiller / leurrer » un grand nombre d'adresses IP

# Problématiques (2/2)



- S Nécessité de déployer plusieurs plate-forme honeynet
  - Q Coût matériel important
  - Q Difficulté de gérer la cohérence d'un parc de honeynet
  
- S Taches d'administration et de supervision importantes
  - Q Besoin en ressources humaines importants
  - Q Compétences pointues requises : rare et cher !



# Comment maximiser la couverture ?



## S Quelques exemples sur le réseau local

### Q Historiquement, *LaBrea*

- Idée : surveiller les adresses IP non utilisées sur le réseau
- Pour ralentir / contrer la propagation de vers et attaques

### Q Autre cas : Honeyd

- Honeypot virtuel permettant d'émuler des centaines de système
- Utilisation d'un démon *ARPD* afin de répondre à toutes les requêtes

## S BUT : maximiser la « couverture »

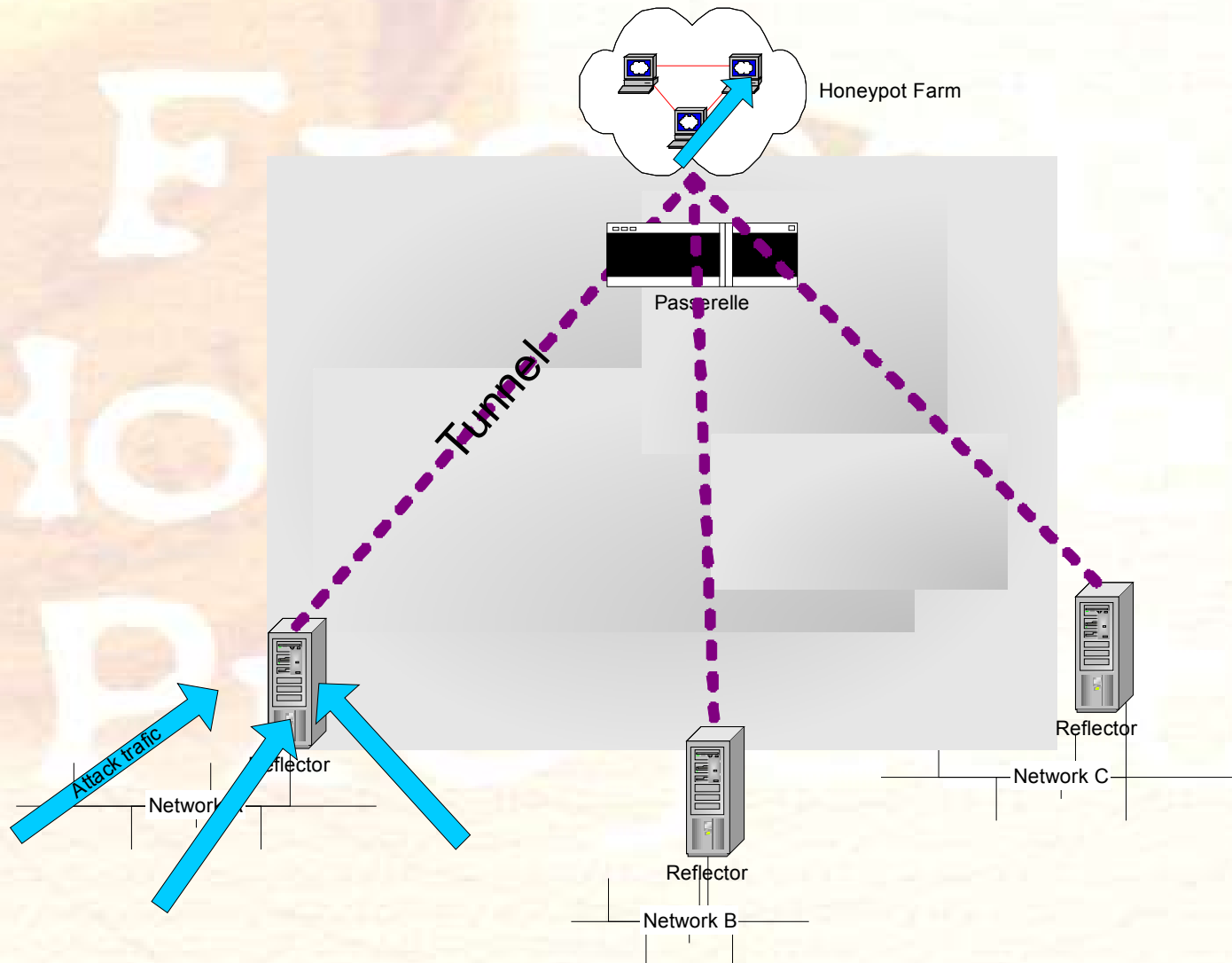
# Le concept de honeypot farms (1/2)



- S Mutualisation des ressources honeypot dans le SOC
  - Q Présence de personnel qualifié dans le *Security Operating Center*
  - Q Administration, supervision H24...
  
- S Utilisation de « relais » honeypot, les « *reflectors* »
  - Q Le reflector émule un système
  - Q Le trafic est alors relayé vers le « honeypot farm »
  - Q De façon transparente

L'idée est de disposer de multiples « reflectors » dans le réseau, et de mutualiser la plate-forme honeypot

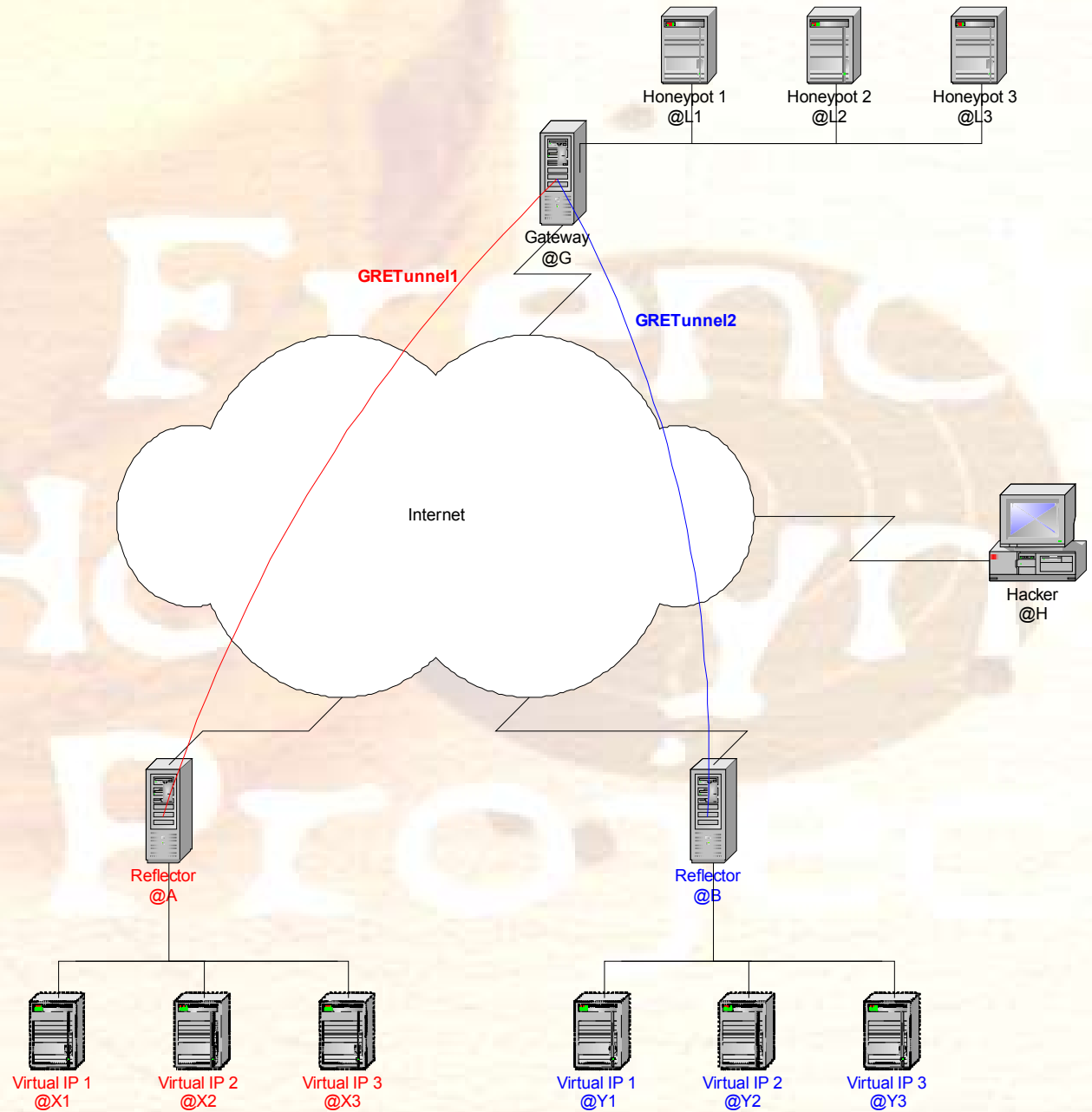
# Le concept de honeypot farms (2/2)



# Exemple d'implémentation



- S** Reflector basé sur un système Linux
  - Q** Capture du trafic local avec un démon ARPD
  - Q** Simulation d'un réseau virtuel
- S** Utilisation de tunnel GRE
  - Q** Relayage transparent des flux vers la ferme de honeypot
- S** Utilisation d'une passerelle Linux
  - Q** Terminaison des tunnels GRE
  - Q** Netfilter pour la gestion des flux
  - Q** IPRoute2 pour le routage des flux
- S** Ferme de Honeypot : Choix libre



# Détail de l'implémentation : les reflectors



- S « proxy arp » pour faire un « puit de trafic »
- S Routage vers la passerelle dans un tunnel GRE
- S Tout le trafic est donc re-routé de façon totalement transparente, vers la ferme
- S Le reflector assure l'acheminement de la réponse vers le hacker

# Détail de l'implémentation : la gateway



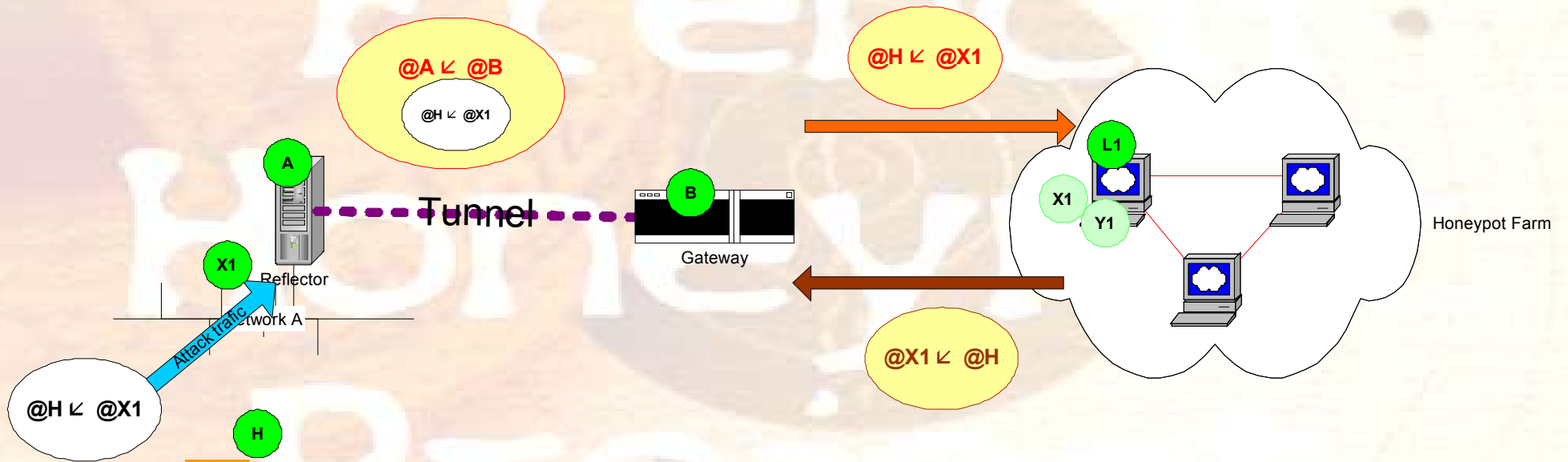
- S Passerelle sur plate-forme Linux
- S Terminaison des tunnels GRE
- S Utilisation d'IPRoute2
  - Q Pour les fonctions de routage avancées
  - Q Routage selon l'adresse source pour le trafic retour
- S Netfilter pour le « packet mangling », c'est-à-dire l'incrémentement du TTL

# Détail de l'implémentation : la ferme

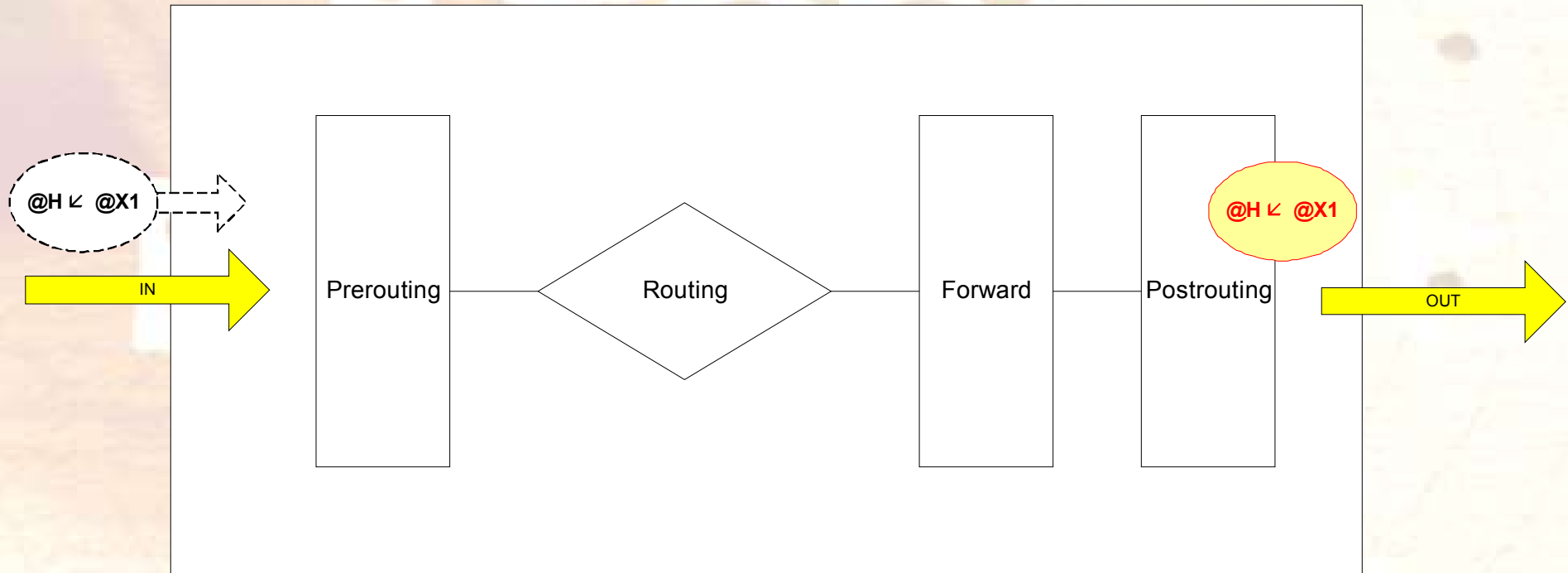


- S Système de type honeynet, constitué de plusieurs machines leurres (plusieurs honeypots)
- S Ceux sont eux qui subissent les attaques des « hackers »
- S Afin de mutualiser les ressources, un système honeypot physique émule autant de système virtuel que nécessaire
  - Q Utilisation d'adresses IP virtuelles multiples sur chaque leurre (honeypot)
  - Q Une adresse est nécessaire par adresse émulée sur le reflector

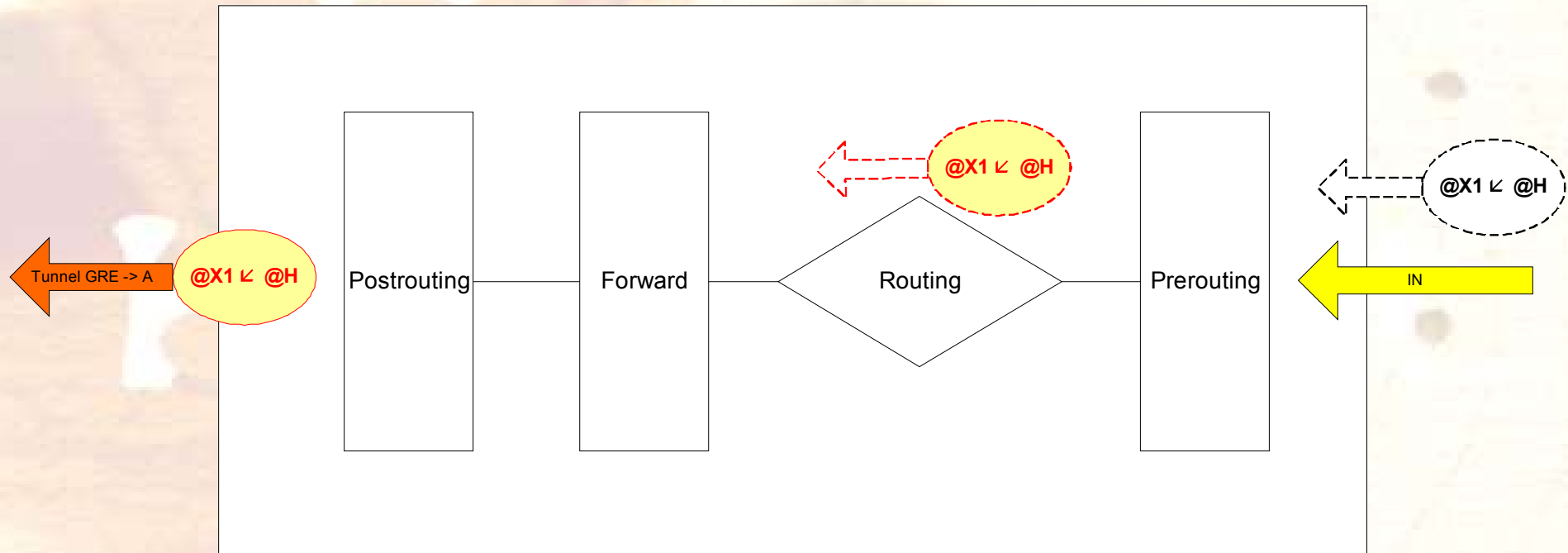
# Principe général



# Fonctionnement de la gateway (1/2)

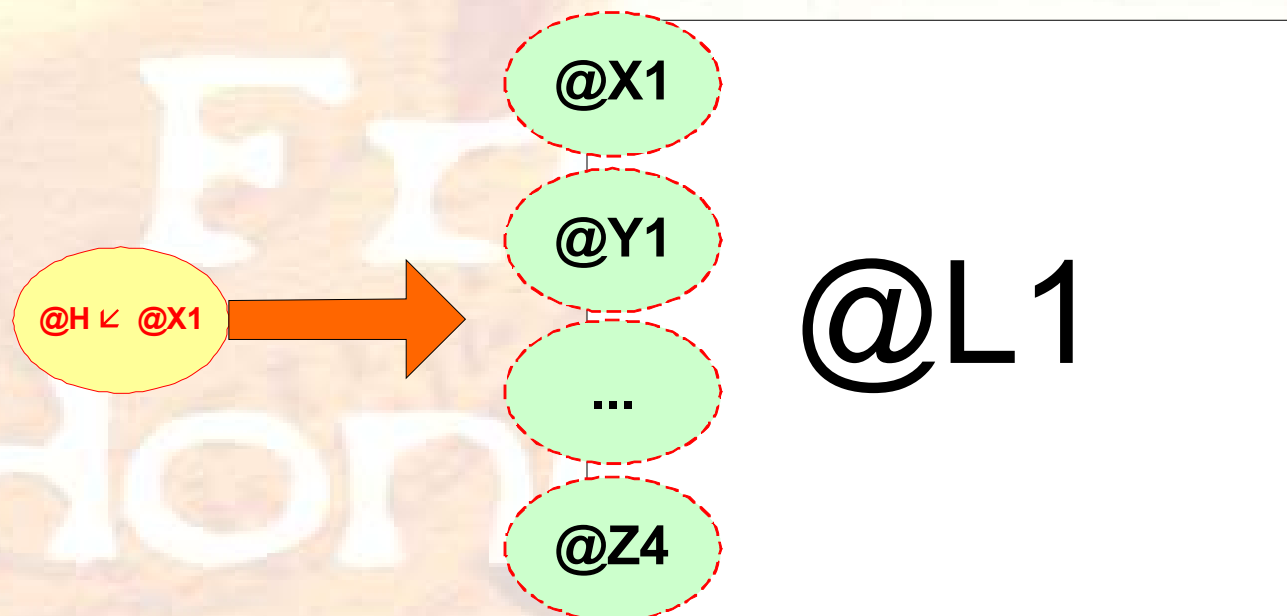


# Fonctionnement de la gateway (2/2)



La gateway route le trafic retour en fonction de l'adresse source du paquet. Le tunnel GRE à destination du reflector A est choisi dans cet exemple, (car X1 est associé à A)

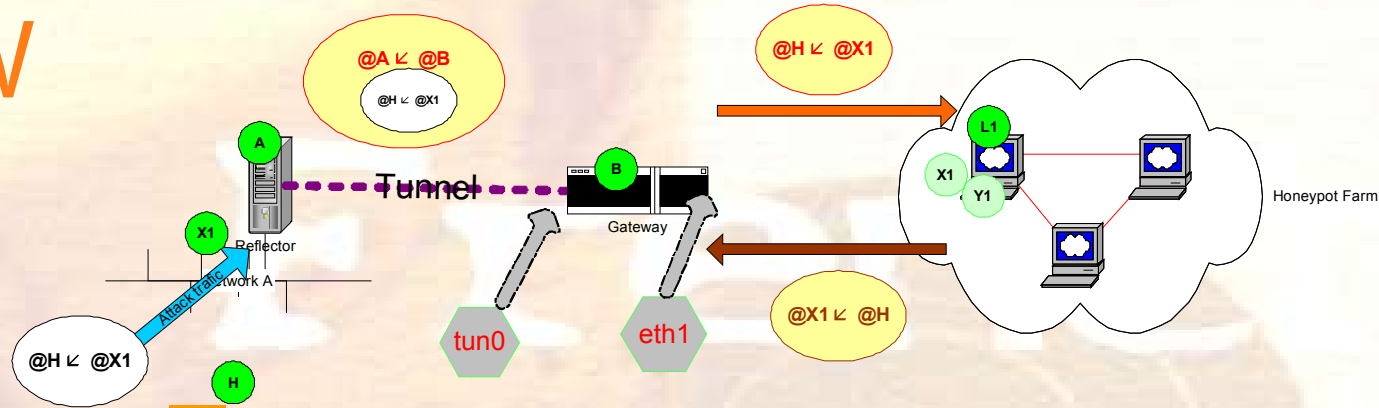
# Fonctionnement des leurres



Le leurre L1 dispose de plusieurs adresses X1, Y1... virtuelles mappées sur ce honeypot suivant l'architecture du système



# Exemple : Table de routage de la GW



## S Avec IPRoute2 (Linux)

Q Paquets entrant la gateway B ( reflector -> HP Farm)

-ip rule add to x1/24 dev tun0 table outTunnel

Q Règle présente dans la table outTunnel

-ip route add x1/24 dev eth1 table outTunnel

Q Paquets retour (HP Farm -> reflector)

-ip rule add from x1/24 dev eth1 table inTunnel0

Q Règle présente dans la table inTunnel0

-ip route add default dev tun0 table inTunnel0



# Quelques points à noter



- S Les adresses virtuelles X1, X2, Y1... sont assignées à deux équipements disjoints, un reflector et un leurre associé
- S Une configuration fine est à réaliser sur la gateway, au niveau de la gestion des tables de routage
  - Q Utilisation de *iproute2* pour le routage retour en fonction de l'adresse source
- S Lors de la définition de l'architecture de cette ferme de honeypot, il faut :
  - Q Réfléchir à la distribution des adresses virtuelles des reflectors (@x1, @x2...) sur les leurres L1, L2
  - Q Gérer les aspects routage sur la gateway

# Fonctionnement transparent



- S L'idée est de relayer le trafic de façon transparente
  - Q Il faut donc que le reflector soit transparent
  - Q Il faut que l'encapsulation GRE soit transparente
  
- S Nécessité de modifier les paquets
  - Q Action « *Mangle* » dans Netfilter
  - Q Incrémentation du TTL (Time To Live) dans les paquets IP au niveau des reflectors
    - Permet de rester invisible aux outils type « *traceroute* »
  
- S D'autres actions seraient nécessaires
  - Q Réflexions en cours... challenge intéressant !

# Conclusions



- S Concept de « Honeypot Farm » très prometteur
  - Q Intéressant de mutualiser les ressources
  - Q Maximisation de la couverture
  - Q Exploitation du honeypot facilitée (SOC?)
  
- S Des améliorations à apporter
  - Q Problème de furtivité du honeypot farm ?
  - Q Quid si honeypot de type « forte interactivité ? »
  
  - Q Mais ce problème ne se pose pas que pour les honeypot farms...