Sécurité autonome dans les réseaux de l'Internet

Philippe Owezarski
LAAS-CNRS, Toulouse, France
owe@laas.fr

Contexte - Objectifs

1. Protéger l'infrastructure de communication et son trafic des attaques / anomalies
   - Préserver le bon fonctionnement, les performances et la QoS (et revenus)
   - Attaques contre les infrastructures de communication (routeurs, switch, …)
     - DoS
     - Intrusions ???
2. Protéger les utilisateurs
   - Service à forte valeur ajoutée

- Temps de réaction à une attaque/anomalie courts
  - ➔Réaction automatique
  - ➔cas des attaques 0d / erreurs jamais rencontrées

Supervision et données collectées

- De nombreux types de données existent :
  - Traffic général contenant des anomalies et des attaques
  - Les filtres de trafic(firewals, IDS, …)
  - Les filtres anti-SPAM
  - Outils de collecte de malware (Virus, worms, …)
  - Traces issues de pots de miel ou des darknets
  - Etc.

➔Comment collecter et utiliser efficacement ces données ?

  - Détection “proactive” d’attaques et classification
  - Estimation du risque
  - Application automatique de contres-mesures

Deux présentations en une…

- Plate-forme d'analyse de malwares
  - Pots de miel, sandbox
  - Firewall automatiquement configurable
- Détection statistique d'anomalies du trafic

Plate-forme pour l'exécution contrôlée de logiciels malveillants

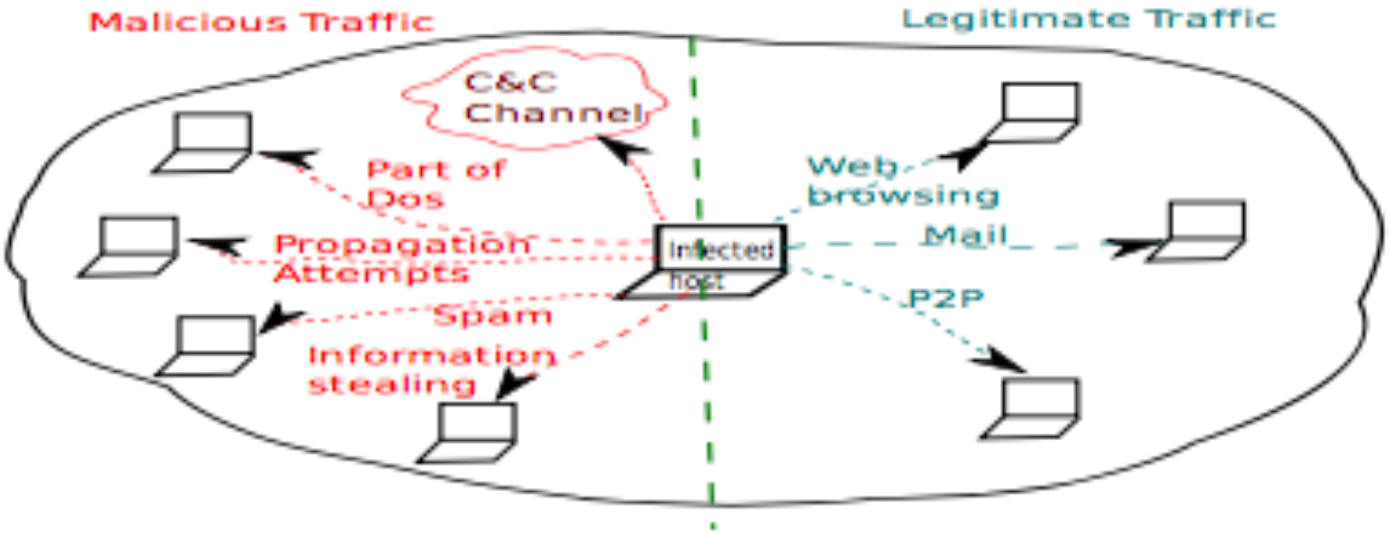
Travail mené avec Ion Alberdi et
Vincent Nicomette

Plan

- Besoin d'observation et analyse du trafic malveillant
- Conception du filtre de paquets
- Implémentation et performance
- Conclusion

Les botnets

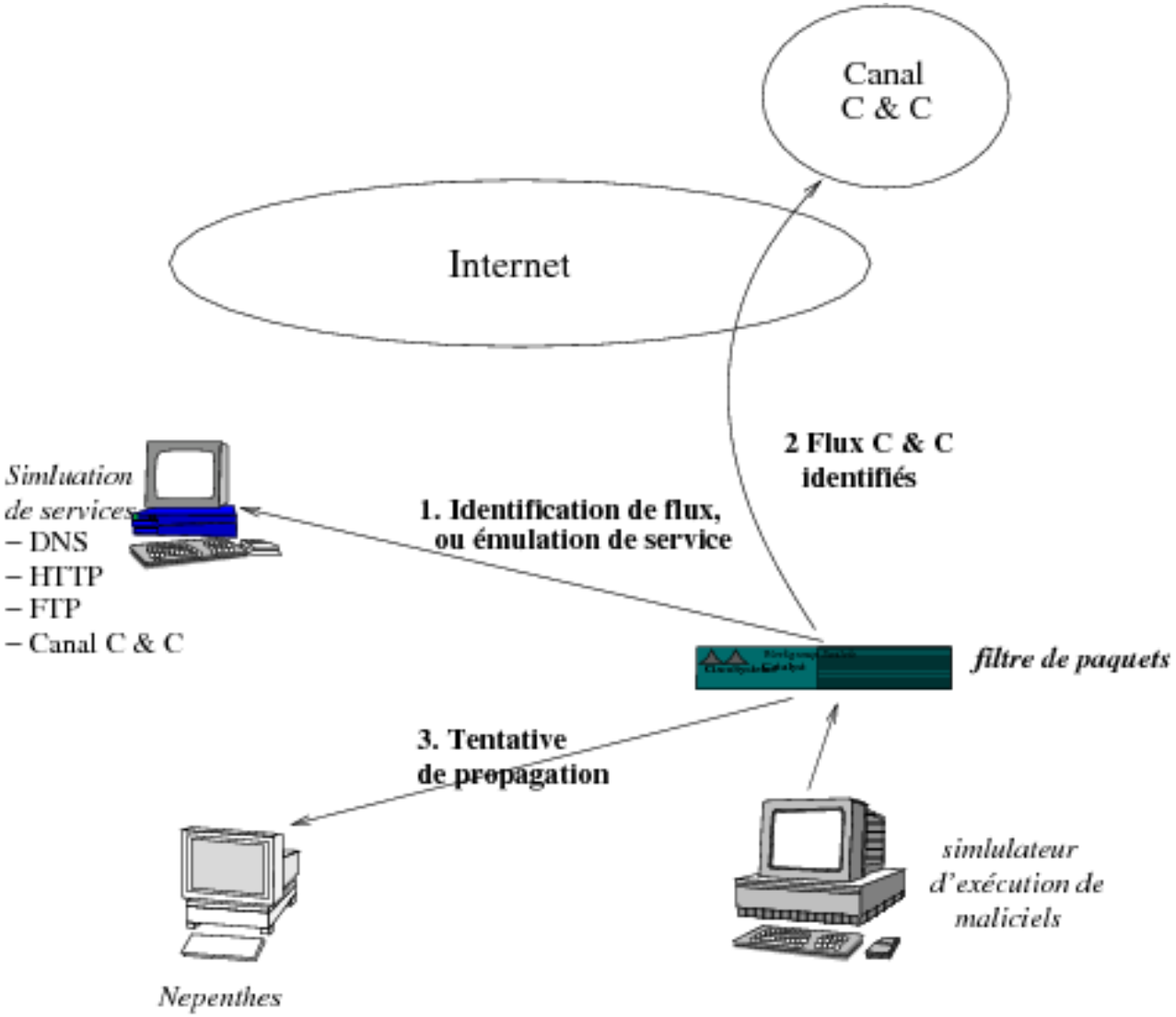
- Les botnets, pilotés à distance par un ou plusieurs maitres, utilisent Internet pour :
  - Se propager
  - Lancer des attaques de déni de service
  - Envoyer du spam
  - Voler des informations confidentielles



Solutions contre les botnets

- Développer des logiciels sans bugs !
- IDS
  - Antivirus → mauvais résultats
  - Supervision du trafic : DoS volume, attaque par simulation de flash crowds ?
- Outils d'observation de botnets, pour la détections comportementale de bots
  - Honeypots : seule la propagation est observée
  - Sandbox :
    - Émulation complète de l'environnement, limité et biaisé
    - Requêtes vers l'Internet (aveuglément) laissées, illégal

Solution : approche hybride



Problématiques du filtrage de paquets

Les protocoles et leurs compositions évoluent constamment. Or un système de filtrage de paquets doit pouvoir permettre à l'utilisateur d'effectuer la sélection de son choix. Or de nos jours :

- Pour vérifier une composition non supportée, il faut attendre la version n+1
- Difficile de configurer le traitement d'une composition partielle
  - composition utilisée IP/TCP/SSL/HTTP

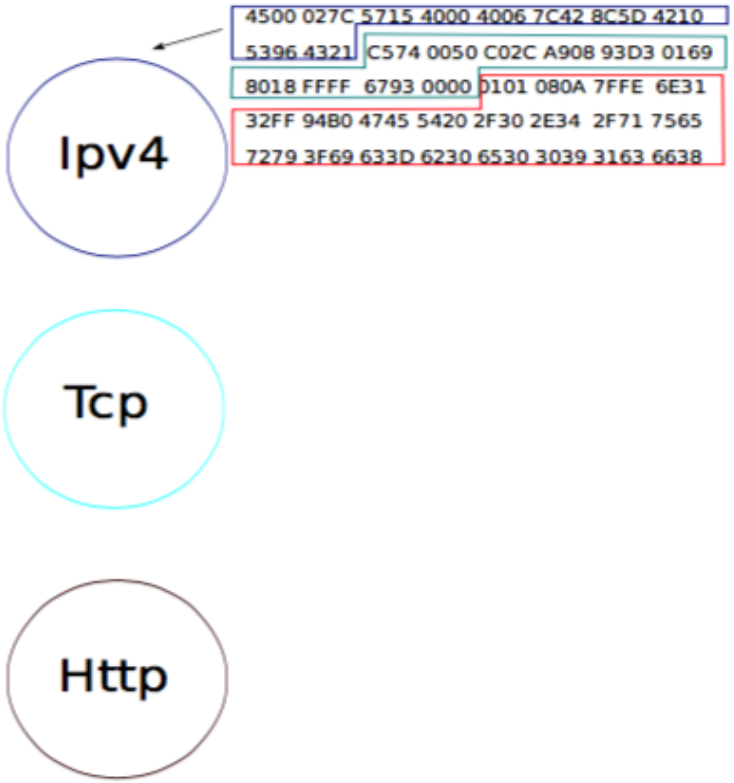
  → Composition inspectée IP/TCP/SSL/HTTP ou IP/TCP' ?

Effets de bord du principe E2E

Extrémité (cas des firewalls)	Milieu
Librairies, tunnels : API Sockets, openSSL, libcurl, IProute, …	Logiciels à compositions monolithiques: solutions clef en main, netfilter, pf, snort, bro, …
IPv4/GRE/IPv4/GRE/ICMP : script shell, Iproute, Ping	IPv4/GRE/IPv4/GRE/ICMP : non configurable
Développement de nouvelles compositions aisé	Besoin de ~tout réimplémenter

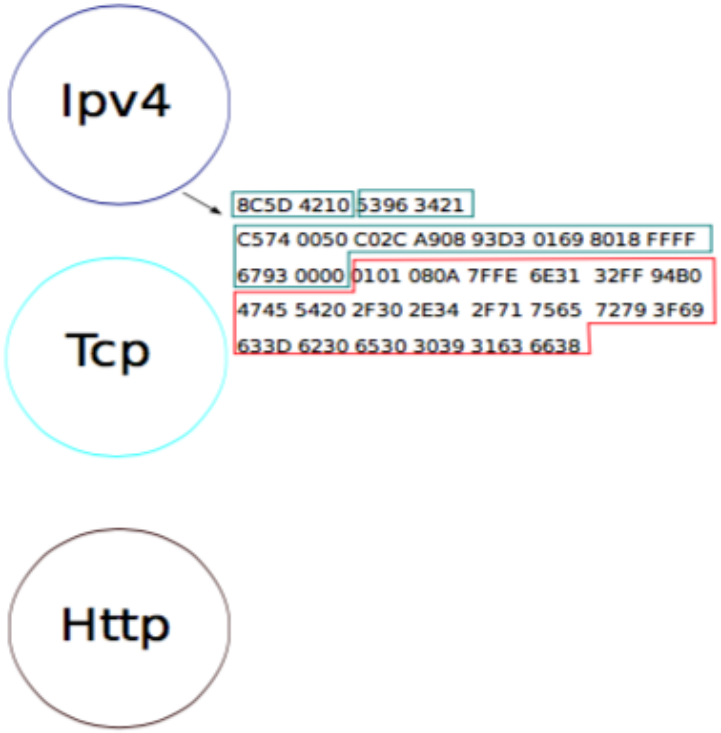
Filtrage classique

Scénario de traitement pour la composition Ipv4/Tcp/Http:



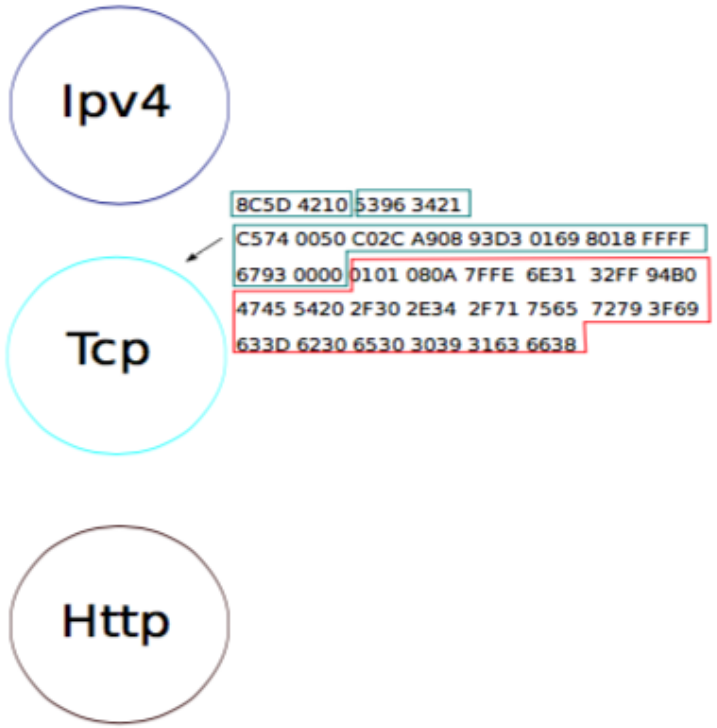
Filtrage classique

Scénario de traitement pour la composition Ipv4/Tcp/Http:



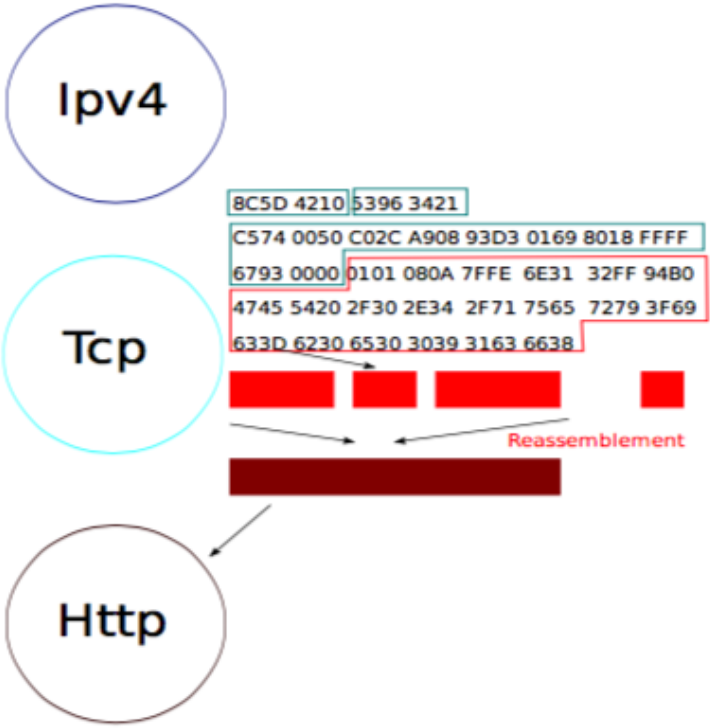
Filtrage classique

Scénario de traitement pour la composition Ipv4/Tcp/Http:



Filtrage classique

Scénario de traitement pour la composition Ipv4/Tcp/Http:



Idée et problématiques

Filtre : arbre d'inspecteurs, ou « mini » pare-feu à états.

Construction de l'arbre à l'aide de deux opérateurs:

- / : composition
- | : parallélisation

Problématiques :

- Correction des compositions : FTP/IP est incorrect
- Agrégation des résultats :TCP/[HTTP | FTP]

Formalisation des types

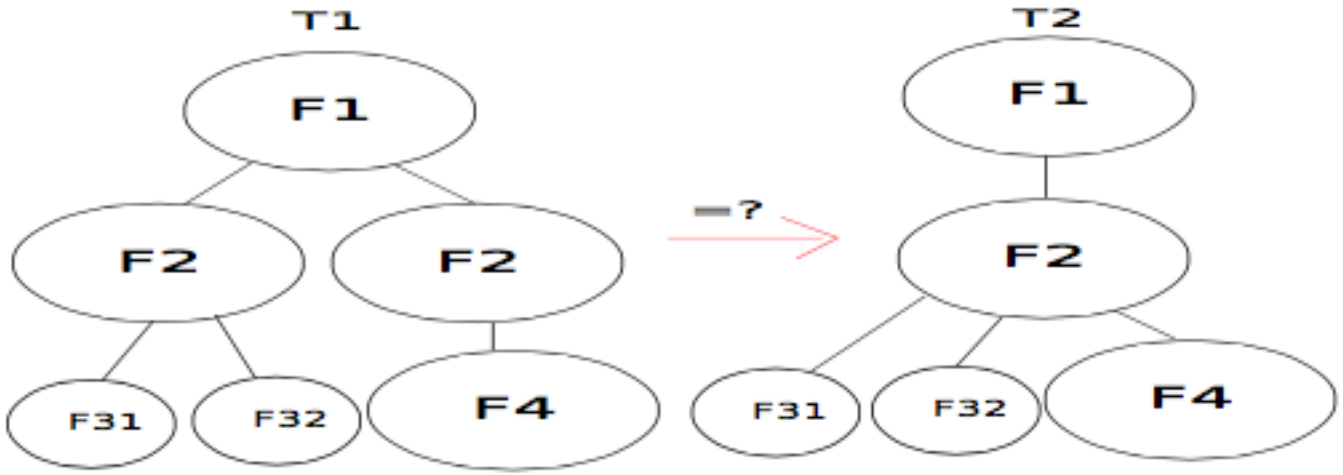
Soit DS l'ensemble des décisions du pare-feu : un « mini » pare feu a :

- te, type paquets en entrée
- ts, type paquets en sortie
- st, type état du « mini » pare-feu
- f : st→te→st x (ts U DS), fonction de filtrage

Il suffit alors de définir :

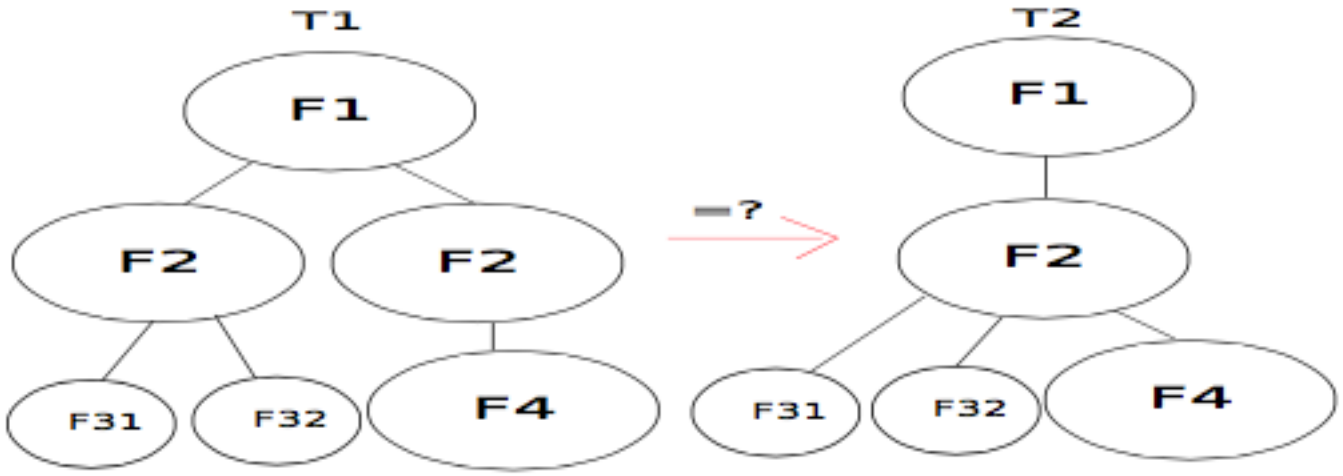
- ST='ST_1 st_1 |…|'ST_n st_n, le type des états de tous les inspecteurs
- PK='PK_1 te_1 |…|'PK_n te_n, le type de tous les paquets
- Ajouter des fonctions from_pk_i : PK→te_i, to_pk_i : ts_i→PK, et définir F qui fait les traductions de format et multiplexe les appels aux f_i

Correction et agrégation



- Correction : arbre valide ssi pour chaque père, pour tous ses enfants, enf_i, $ts_{pere} = te_{enfi}$
- Agrégation : définition du consensus c : P(DS)→DS, c({c({ds_{31}, ds_{32}}), c({ds_4})})(1) = c({ds_{31}, ds_{32}, ds_4})(2) ?

Consensus



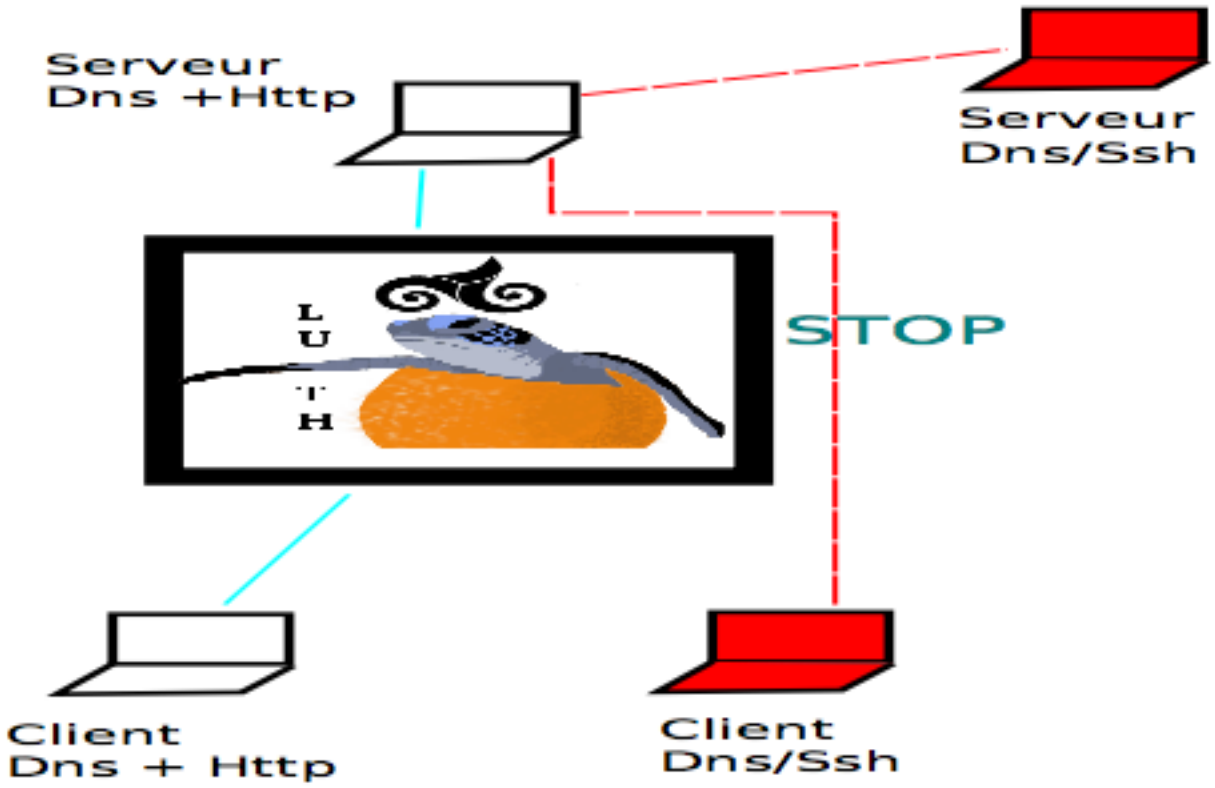
- Contre exemple : c({ds_{31}, ds_{32}}) = ds_4, c({ds_4})= ds_4 ➔
  (1) c({c({ds_{31}, ds_{32}}), c({ds_4})})=ds_4
  (2) c({ds_{31}, ds_{32}, ds_4})=ds_{31} ➔ (1)≠(2)
- $\forall X \in P(P(DS)), c(\bigcup_{x \in X} \{c(x)\}) = c(\bigcup_{x \in X} x)$ ➔ T1 T2 équivalents

LUTH

- Lur Ur ta Haize (pare-feu : Terre Eau et Vent)
- Plus grande tortue des mers : « Rien ne sert de courir, il faut partir à point »
- Implémentation en OCaml (typage fort + garbage collector efficace), en espace utilisateur en utilisant la librairie libnetfilter_queue sous linux
- Pourquoi pas en C ?
  - Tous les pare-feux qui traitent des données applicatives développés en C/C++ ont eu des vulnérabilités critiques

Cas d'étude

- Canal caché DNS : liberté sémantique offerte par les requêtes TXT utilisées pour tunneler divers protocoles dans des requêtes DNS



Deux configurations

- Dans le cas des ponts d'accès WIFI-payants, vérifier les paquets TCP ne sert à rien, le point d'accès les redirige vers un portail pour donner la possibilité de payer l'accès. On a donc testé LUTH sous ces deux configurations:

1. IPv4 (cksum=no)/ [TCP(tcp_only=yes) | UDP(cksum=no)/DNS]);;

2. IPv4 (cksum=no)/ [TCP/TPort(sr_port=80)/Void | UDP/UPort(sr_port=53)/DNS]);

Performances

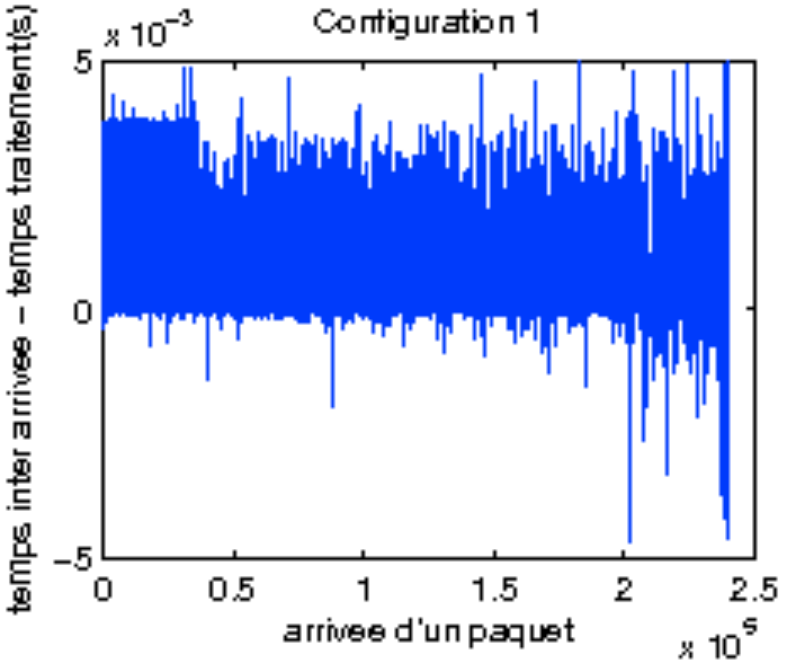
- Le serveur HTTP héberge des fichiers allant de 10 ko à 500ko avec une distribution Pareto de 1.2 (queue lourde)
- Le serveur DNS/SSH complice héberge des fichiers allant de 10 ko à 189ko avec une distribution Pareto de 1.2 (queue lourde)
- Les clients téléchargent un fichier aléatoirement toutes les 10 ms

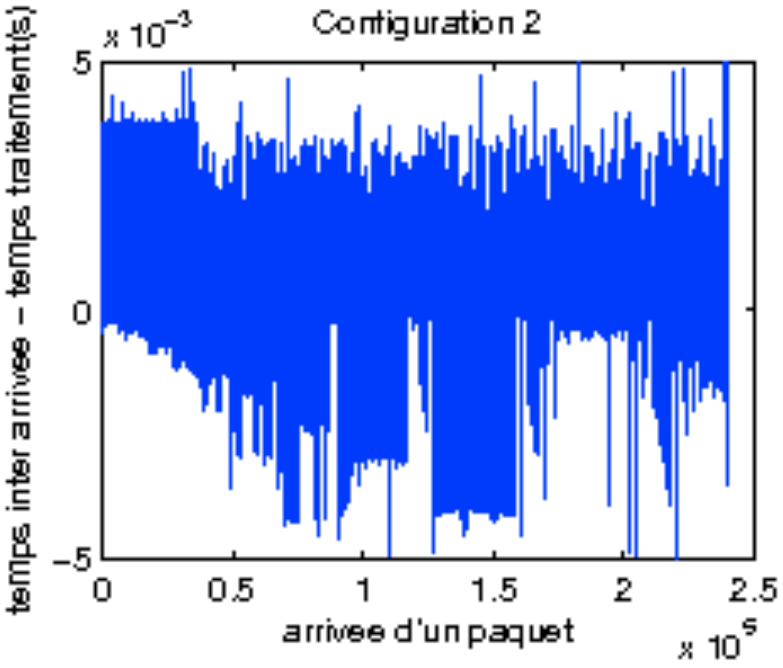
- Filtrage correct dans les 2 cas

Sans fw	Ref	Conf 1	Conf 2
1.112 Mo/s	0.870 Mo/s	0.753 Mo/s	O.692 Mo/s
1.924 Mo/s	0,688 Mo/s	0.582 Mo/s	0.707 Mo/s

Performances (2)

- Analyse hors ligne : consommation mémoire conf 1 ≈10x consommation mémoire conf 2





- Figure : performances du filtre de paquets hors ligne selon les configurations 1 et 2

Conclusion

- Le firewall est configurable en fonction du trafic qui le traverse et qu'il doit analyser

MAIS

- Beaucoup de choses doivent être faites à la main
- La connaissance sur les malware et attaques associées est très parcelaire

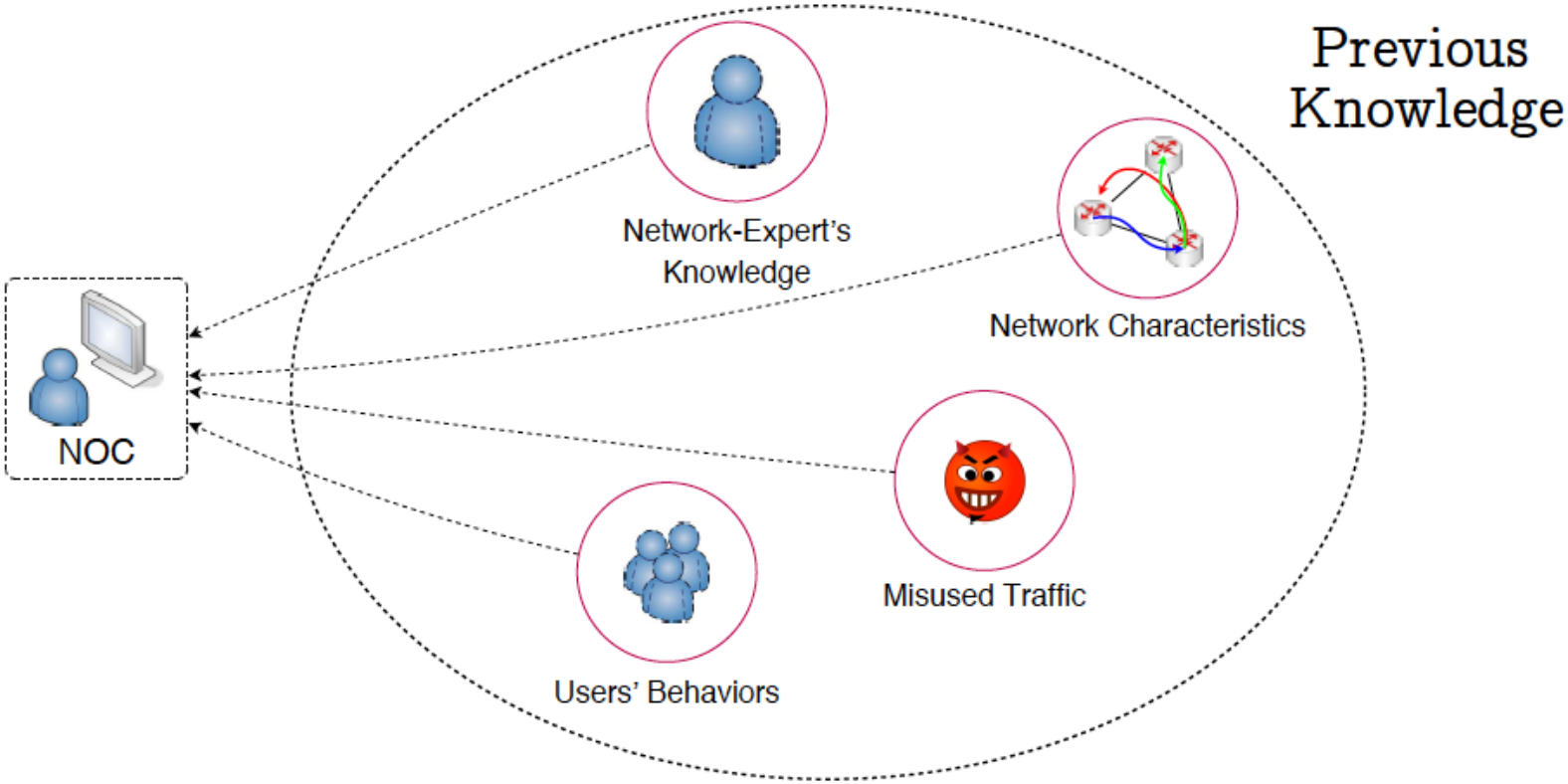
Unsupervised network anomaly detection

With Johan Mazel and Pedro casas

Big security data analysis

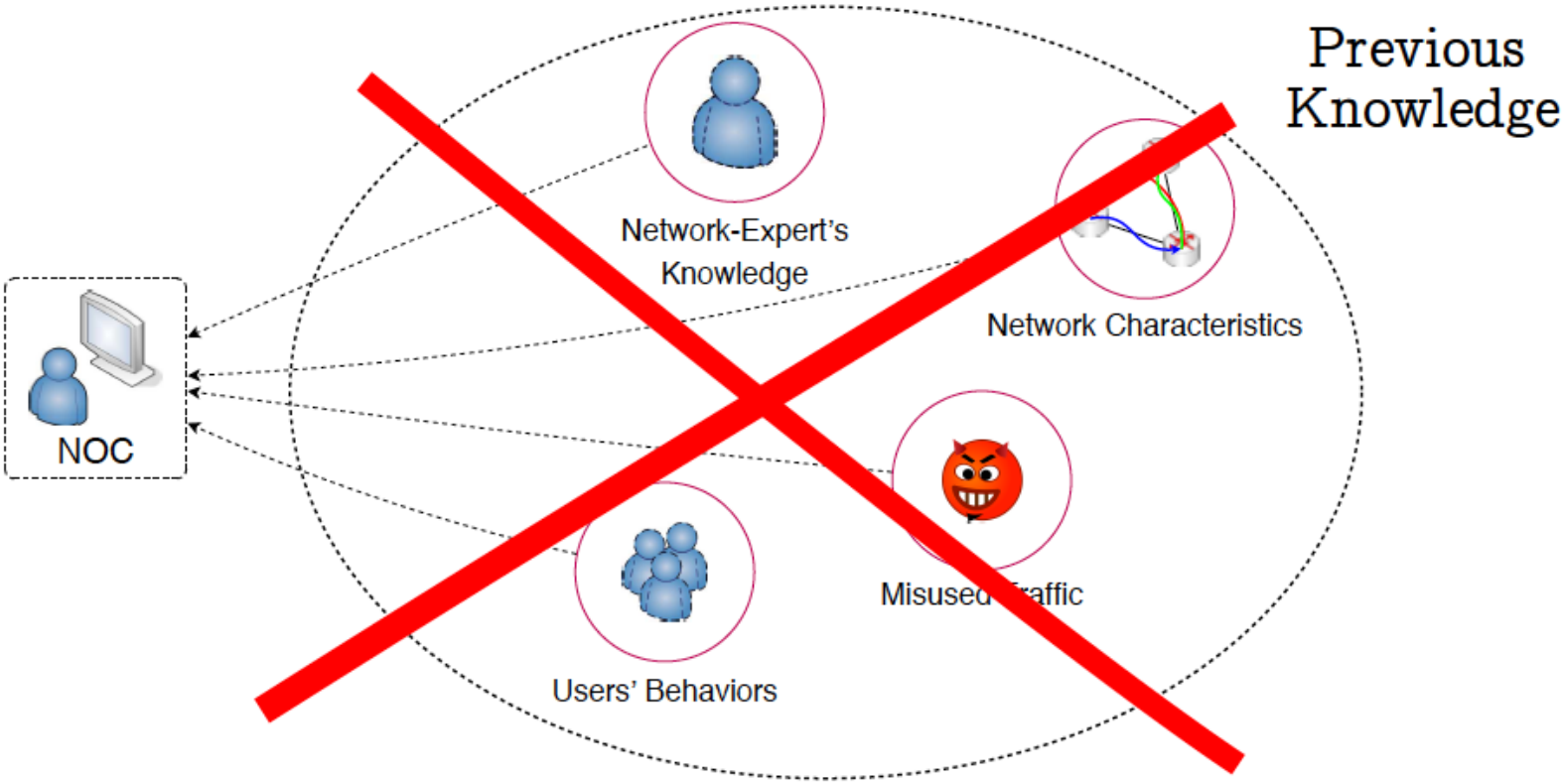
- Strong requirement for autonomous data analysis
  - Unsupervised data mining
  - Unsupervised machine learning
  - Unsupervised characterization of malicious activity
  - ***(Unsupervised classification*** – ***root cause analysis*** *of malicious activity)*
- Can we make characterization and classification unsupervised?
  - Can we avoid relying on the **skills** of a **security expert**?

Towards Autonomous Network traffic Monitoring



However, current monitoring systems highly rely on previous knowledge, which is costly, slowly produced, and difficult to obtain

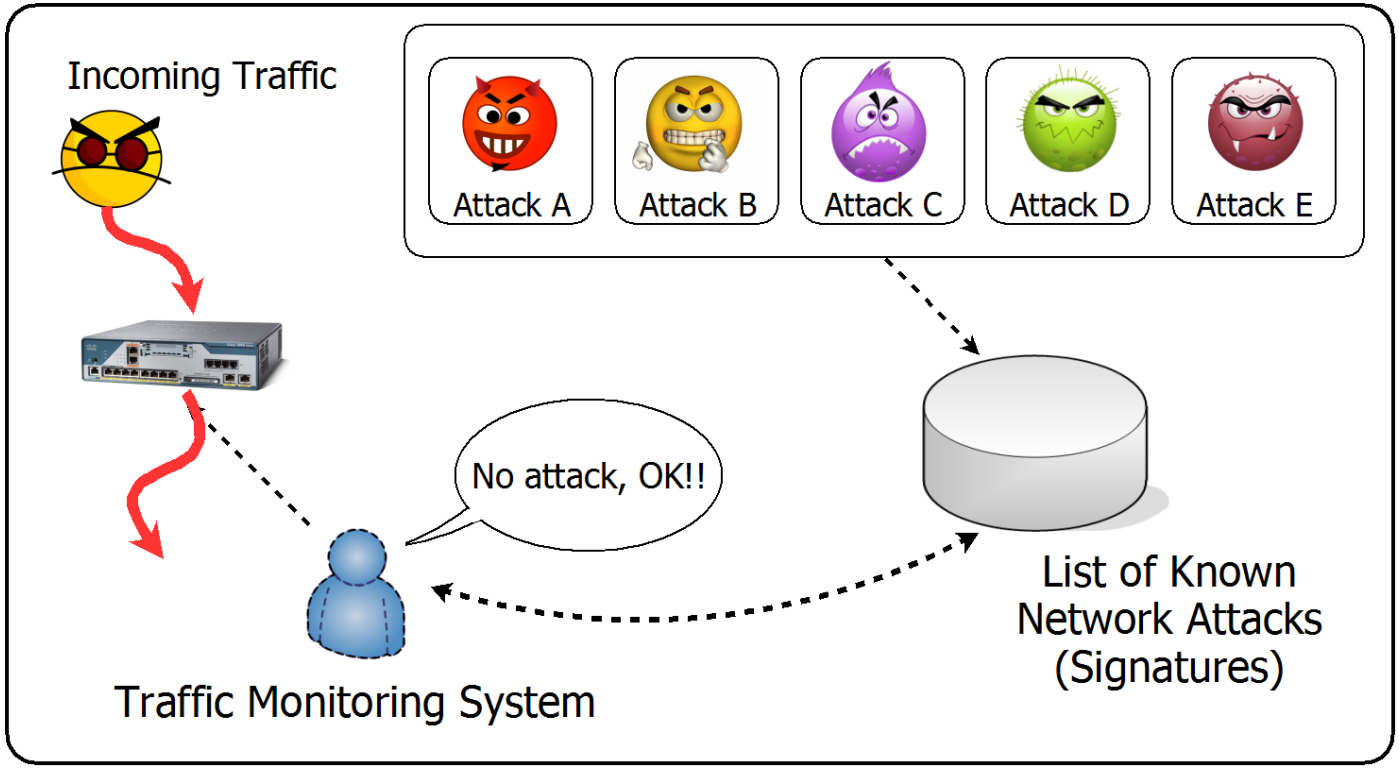
Towards Autonomous Network traffic Monitoring



Can we REDUCE this DEPENDENCE on KNOWLEDGE?

Signature-based AD

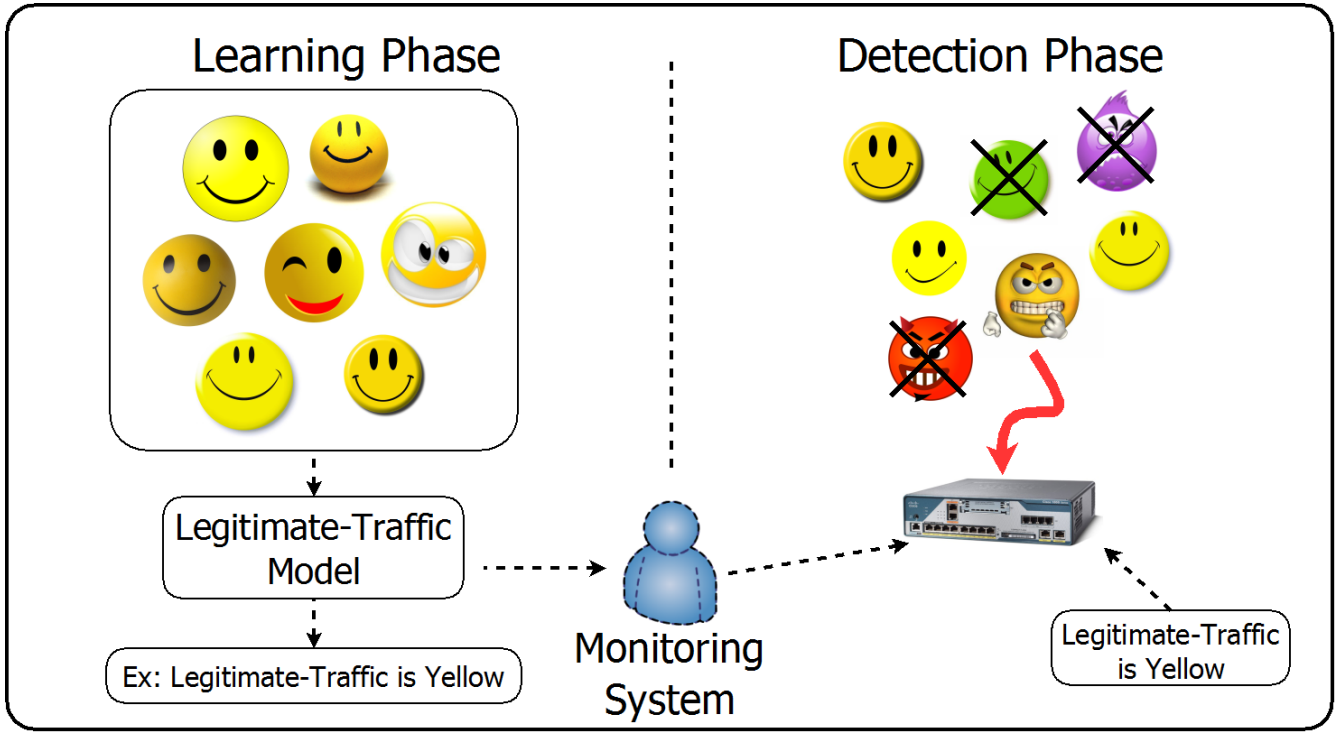
- Detect WHAT I ALREADY KNOW



(+) Highly effective to detect what it is programmed to alert on
(-) Cannot defend the network against unknown attacks
(-) Signatures are expensive to produce: human manual inspection

Supervised-learning-based AD

- Detect what is different from WHAT I KNOW



(+) It can detect new anomalies out-of the baseline
(-) Requires training on anomaly-free traffic
(-) Robust and adaptive models are difficult to conceive

Shortcomings in nowadays network security

Network security is based on some PREVIOUS KNOWLEDGE:

- Signature-based: detect the attacks THAT WE KNOW
- Anomaly detection: detect DIFFERENCES from WHAT WE KNOW

HOW STABLE-in-time is this PREVIOUS KNOWLEDGE?

- Network attacks are a moving target: new attacks are constantly emerging, and the birth-rate is increasing
- New services and applications modify normal-operation profiles

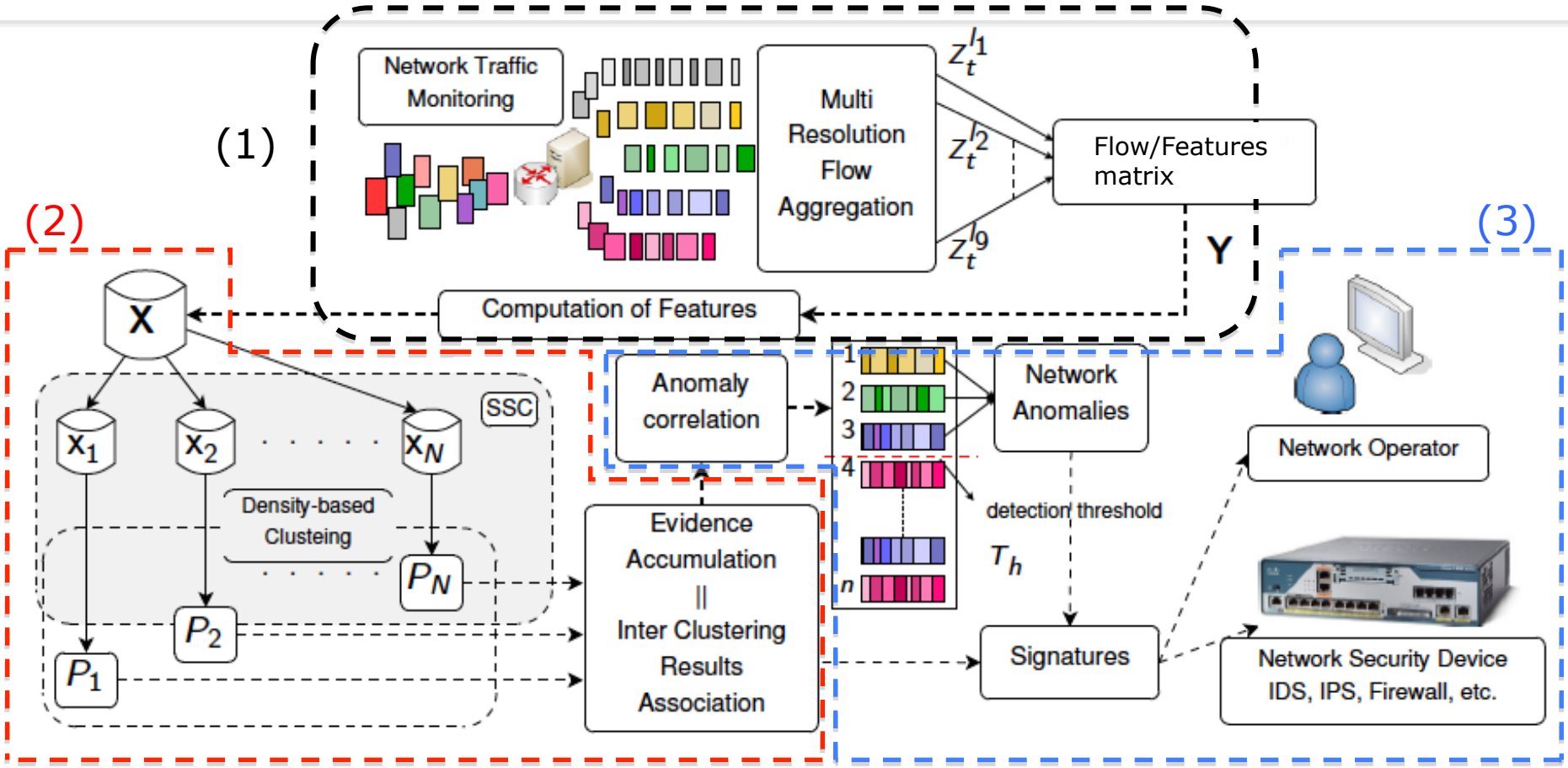
We depend TOO-MUCH on the PREVIOUS KNOWLEDGE:

- This knowledge is difficult and expensive to obtain
- Long periods of VULNERABILITY (e.g. weeks) between a new attack and the construction of a new signature
- Current network security is REACTIVE, and as such WE ARE ALWAYS ONE STEP BEHIND THE ATTACKERS!!!

Our new approach for security

- Unsupervised clustering for detecting and characterizing classes of anomalies without relying on previous knowledge, signatures, statistical training or labeled traffic
- Automatic production of filtering rules (→ firewalls, filtering equipments, …)
- *(Discrimination between legitimate vs. Illegitimate anomalies*
  - *Root cause analysis)*
- Automatic mitigation of attacks vs. Reporting to network/security administrator

Unsupervised network anomaly detection and characterization



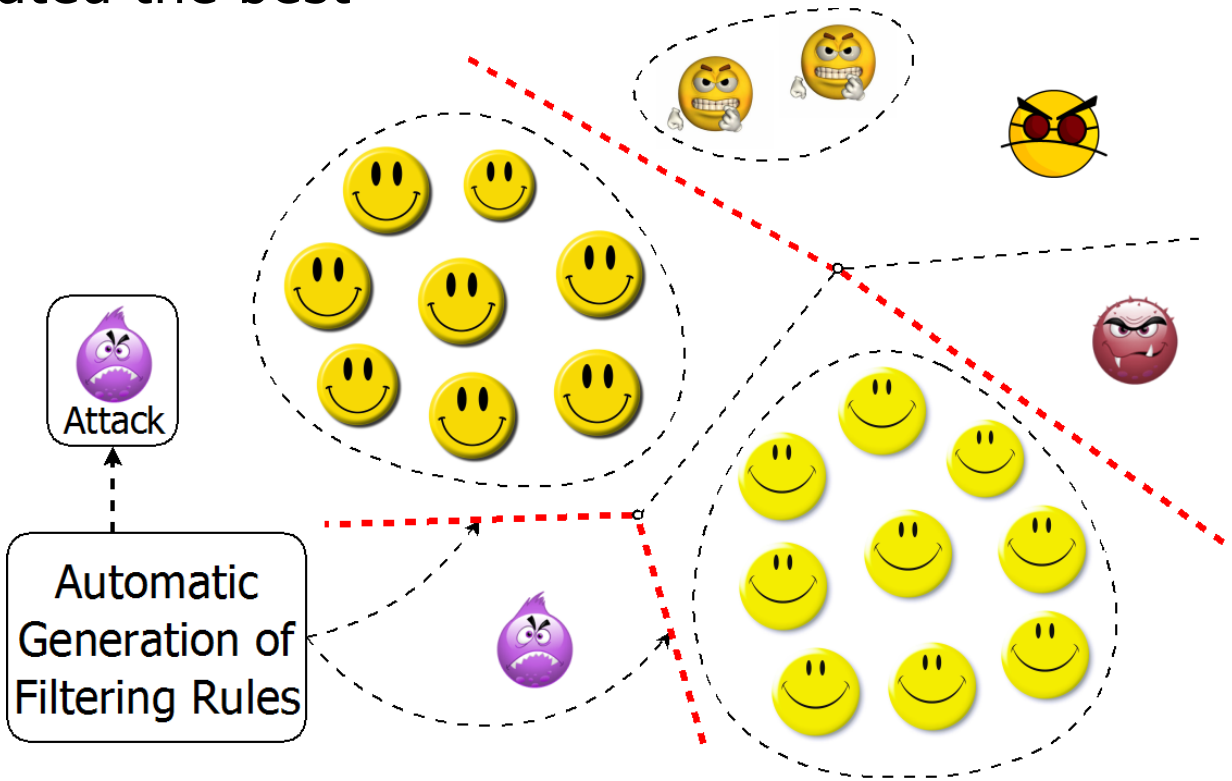
(1) Multi-reso., flow aggreg., Change-detection & Attribute building

(2) Sub-Space Clustering and, evidence accumulation or Inter-Clustering Results Association

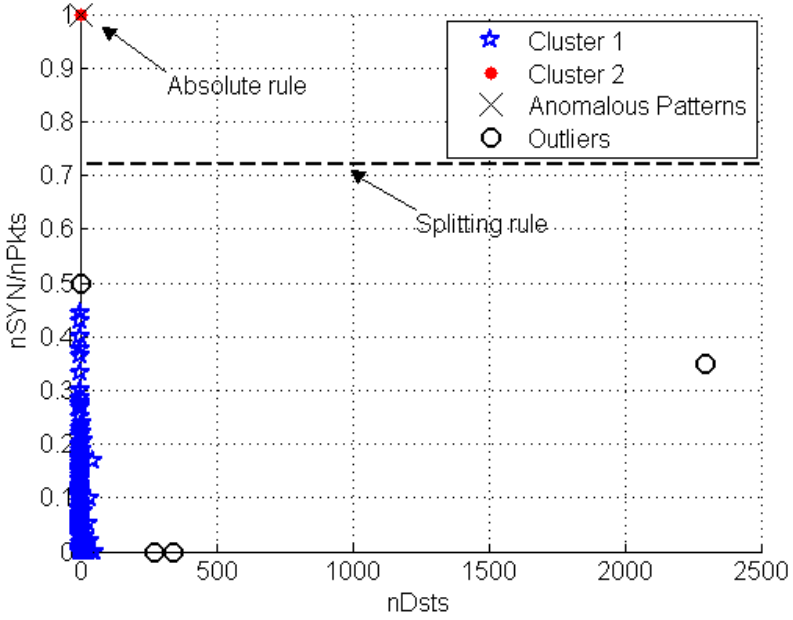
(3) Correlation & Characterization through filtering rules → signatures

Filtering rules for anomaly characterization

- Automatically produce a set of filtering rules f(Y) to correctly isolate and characterize detected anomalous flows
- Select the "best" features to construct a signature of the anomaly, combining the top-K filtering rules
- is isolated the best



detection of a SYN Distributed Denial of Service (DDoS) attack in MAWI traffic



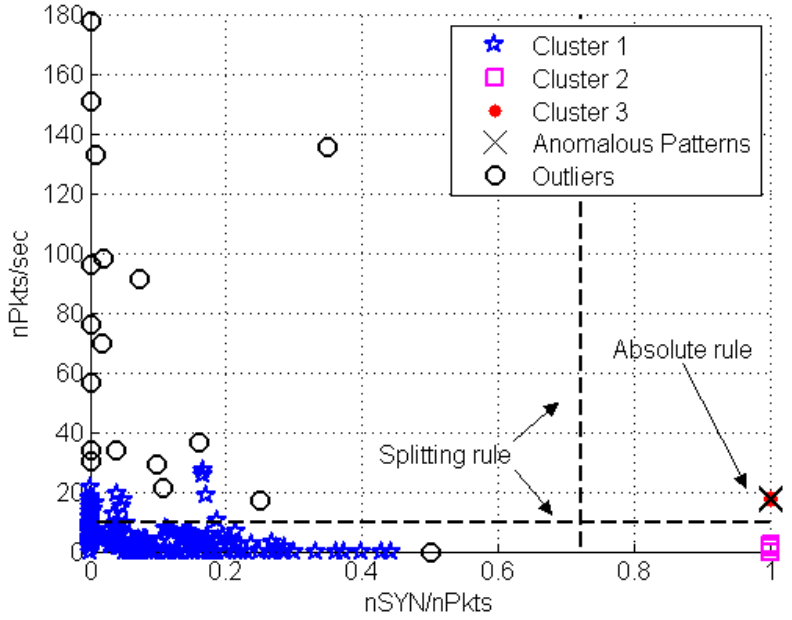


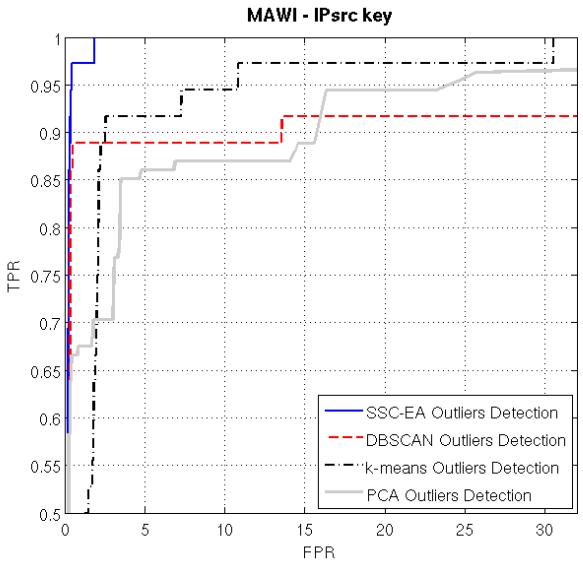
Illustration of clustering graphical results

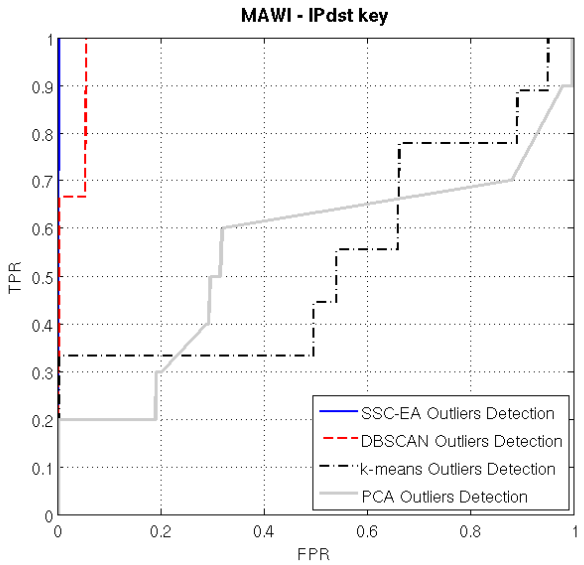
(a) SYN DDoS (1/2) (b) SYN DDoS (2/2)

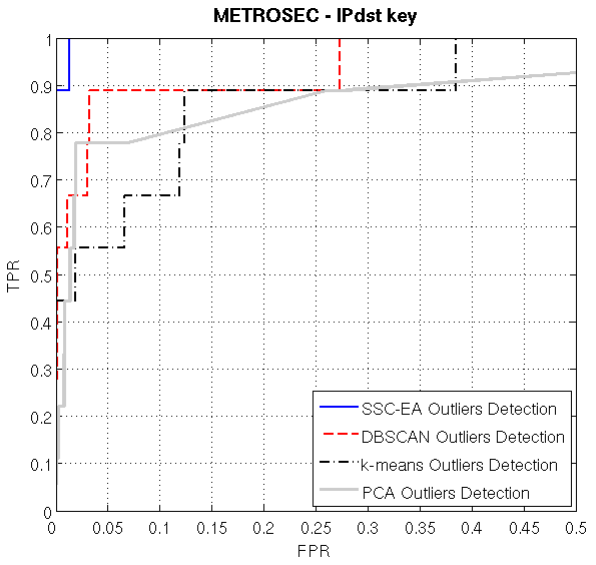
Generated signature

$(nDsts == 1) \wedge (nSYN/nPkts > \lambda_3) \wedge (nPkts/sec > \lambda_4) \wedge (nSrcs > \lambda_5)$

Comparison between ≠ unsupervised techniques





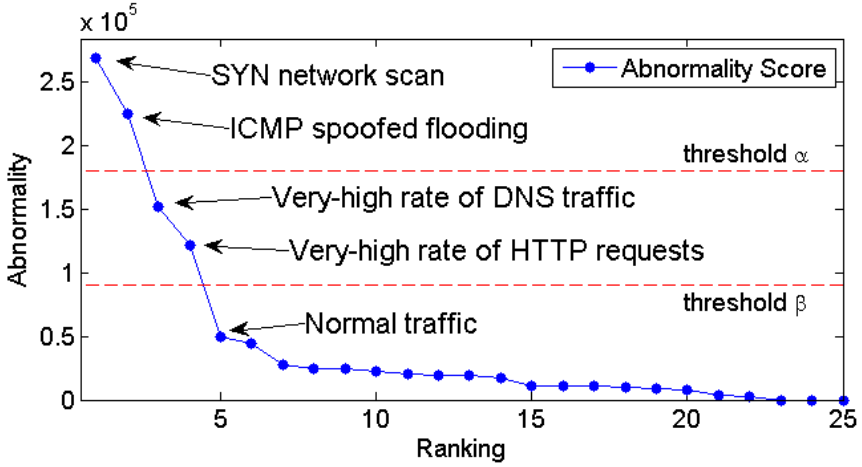


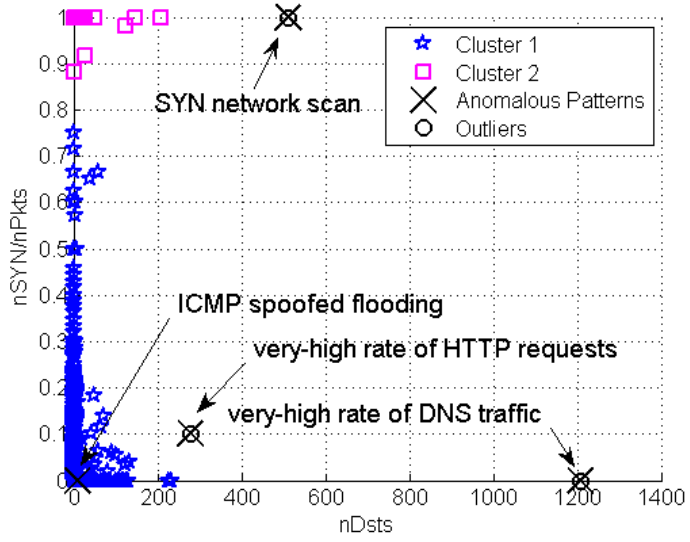
Comparison of detection performance of several detection algorithms

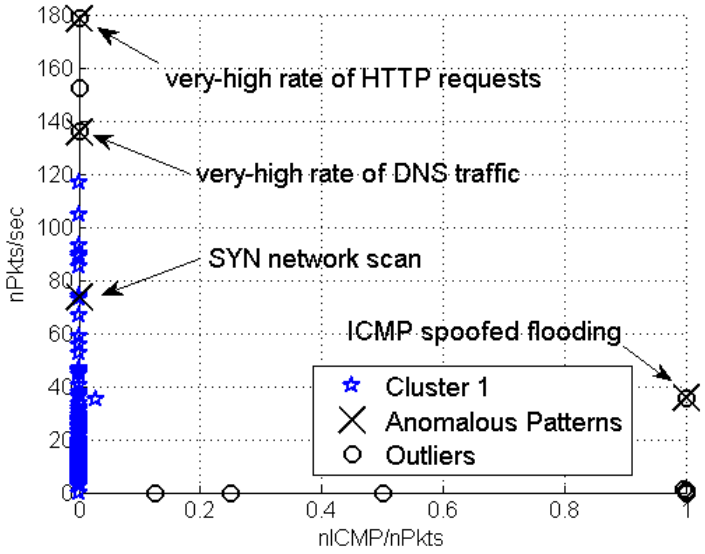
ROC (receiver Operating Characteristic) curves presenting True Positive Rate (TPR) vs. False positive rate (TPR)

Attacks detection & characterization in MAWI traffic

- Detect network attacks that are not the biggest elephant flows







Conclusion

- Detection / classification reports of anomalies
- Reports are very complete in order to allow the automatic enforcement of countermeasures for the ML engine

(+) filtering rules ready to be exported towards security devices (e.g. Intrusion Detection Systems, Intrusion Protection Systems, Firewal, etc.)

➔ We cannot become completely independent of a previous knowledge, but unsupervised / semi-supervised ML make a crucial building-block for self-defense

Future work

- Traffic classification
  - Root cause analysis ? What is the intention behind an anomaly ?

  → Statistical approach ???

  → Based on active monitoring
    - (exploiting the principle of the malware replay platform)

- ???

That’s all folks !