# FIELDS : Flow Intrusion Extrusion Largescale Detection System

Nicolas Greneche

# FIELDS : Flow Intrusion Extrusion Largescale Detection System

**Nicolas Geneche**
DSI – LIPN / Université Paris 13
99 Avenue Jean Baptiste Clément
93 430

## Résumé

*Le NSM (Network Security Monitoring) est une supervision des interactions réseaux orientée sécurité. Cette surveillance est basée sur les traces collectées au niveau réseau. Quatre niveaux de collecte ont été identifiés par ordre croissant d'espace occupé : statistiques très générales (répartition de la quantité de données par protocoles, nombres de connexions TCP), flux (qui a parlé avec qui, comment, quelle quantité de données échangées), alertes, essentiellement desremontées de NIDS (Network Intrusion Detection Systems) et enfin la capture exhaustive.*

*FIELDS se place entre le flux et la capture exhaustive. Étant donné que la plupart des malwares modernes chiffrent leurs communications, l'analyse de la charge utile est inefficace. Cependant, avoir une capture exhaustive des communications d'une machine infectée (ou suspectée) peut être intéressant dans une optique d'analyse post-mortem réseau. Packet Filter, le pare-feu de la famille BSD, propose deux fonctions intéressantes. Les tables qui sont des listes d'adresses IP optimisées pour répondre à la question "est ce que telle IP est dans la table ?" et une possibilité de capturer au format PCAP les paquets réseaux matchant une règle. FIELDS est un développement se basant sur PF permettant d'ajouter l'IP source et / ou destination dans une table lorsque le paquet coïncide avec une règle. Ces différentes règles modélisent des communication suspectes (blacklist, sinkhole IP etc.). Une fois l'IP suspecte ajoutée dans une table, une règle finale est ajoutée pour capturer les communications provenant et / ou à destination de cette IP. Cette capture peut ensuite être traitée par des IDS pour confirmer / infirmer la compromissionou archivée soit pour de l'analyse post-mortem réseau soit pour introduire un niveau supplémentaire de conservation de flux.*

## Mots-clefs

*Flux réseaux, Détection d'intrusions, Analyse post-mortem réseau, Capture réseau*

## 1 Introduction

Les systèmes de détection d'intrusions classiques sont connus depuis longtemps [1]. Ils opèrent à la fois sur le trafic entrant et sortant avec des approches soit par signatures (Snort), soit par une analyse comportementale (essentiellement la conformité des interactions réseaux par rapport aux protocoles annoncés, comme dans Bro). Le travail autour de cette discipline se concentre sur l'apprentissage [2] ou la modélisation [3] de comportements malveillants. L'inconvénient principal est qu'étant donné la quantité phénoménale de trafic générée à l'échelle d'une université (environ 900 Mo de PCAP pour stocker 45 secondes) il faut déjà disposer de matériel suffisamment performant pour tenir la charge et aussi de temps humain pour analyser les alertes. FIELDS intervient à ce niveau en utilisant des heuristiques afin de réduire le volume de données à analyser. Ce pré-traitement a deux finalités : réduire la quantité de données à traiter pour les NIDS et / ou augmenter la durée de rétention des événements réseau en ne sélectionnant que certaines parties du trafic à archiver.

## 2 État de l'art

FIELDS s'appuie le pare-feu PF (Packet Filter) issu du monde BSD pour modéliser des heuristiques de capture de paquets. Cette section va présenter les différentes heuristiques existantes et motiver le choix de PF comme socle pour FIELDS.

## 2.1 Heuristiques

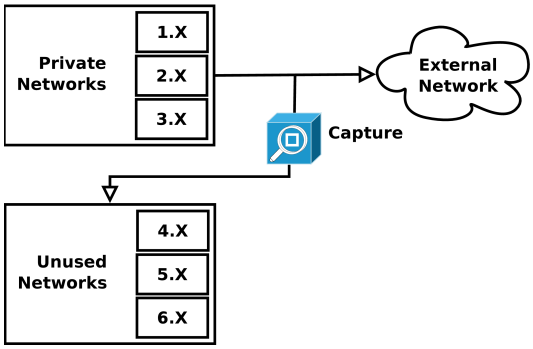


*Figure 1 - Sinkhole IP*

Les heuristiques sont des motifs de communications réseaux. Pour modéliser ces motifs, il est nécessaire de disposer d'un langage. Un pare-feu intègre tout un langage capable de modéliser des flux malveillants (son objectif premier étant de les bloquer). De plus, les implémentations modernes de pare-feu libres intègrent des proxys pour réaliser des opérations de niveau supérieur à la couche 4 du modèle OSI (par exemple pour filtrer du FTP). Le trafic correspondant à un motif est sélectionné. Nous allons voir trois types d'heuristiques :

- Blacklist : Le type le plus simple. Une liste d'adresses est maintenue. Tout paquet à destination d'une de ces adresses est considéré comme suspect. Dans cette liste on peut trouver des Honeypots, des adresses récupérées par rétro analyse de malwares (celle ci sont parfois codées en dur) etc. Cette méthode est totalement inefficace en cas d'utilisation de méthodes basées sur le Fast Flux [4], l'objectif étant justement de cacher la destination réelle du trafic ;
- DNS Blacklist : Une liste de noms de domaines suspect est maintenue. Quand une machine du réseau demande une résolution sur un des domaines de la liste, son adresse est sélectionnée. Ce type pourrait être implémenté sous forme de proxy opérant sur les requêtes A. Dans un environnement expérimental de rétro analyse de malwares, une adresse de Honeypot peut être fournie [5] ;
- IP Sinkhole : Historiquement, le sinkhole est utilisé par les FAI au niveau BGP pour journaliser et / ou rediriger les attaques contre leurs clients [6]. Cette méthode est également applicable aux réseaux IP. L'administrateur connaît toutes les plages privées (au sens RFC 1918). Un paquet provenant d'une machine du réseau à destination d'une adresse privée qui n'est pas dans la liste des plages privées du site est suspect. Pour récupérer les paquets à destination de ces réseaux non utilisés, le trafic vers ceux ci est routé vers un routeur instrumenté qui logue les paquets. On ne récupère donc que les paquets à destination du sinkhole [7].

Les méthodes présentées sont clairement orientées extrusion [8]. Une extrusion est un comportement illicite provenant d'une (ou plusieurs) machine(s) du réseau interne vers l'extérieur. Une extrusion est donc la conséquence d'une intrusion réussie.

## 2.2 Packet Filter (PF)

PF est le pare-feu intégré aux systèmes [Free|Open]BSD. PF dispose de deux fonctions et d'une implémentation particulièrement intéressantes pour FIELDS. Les deux fonctions sont les tables et la journalisation de paquets sous forme de PCAP. Les tables sont des listes d'IP utilisées dans le jeu de règles. L'intérêt de ces listes est que le temps de latence pour répondre à la question « telle IP fait elle partie de la liste ? » est très court. Ces listes sont consultables et modifiables depuis l'espace utilisateur. Au niveau noyau, on peut ajouter des IP dans les tables sur la base de règles statistiques. Une application courante est de bloquer les bruteforce sur SSH. Au delà d'un certain seuil de connexion pour une IP sur un serveur SSH, l'IP distante est ajoutée dans une table contenant les attaquants bruteforce.

PF propose également de journaliser les paquets coïncidant avec une règle. Les paquets sont stockés au format PCAP. Enfin les proxy / modules pour analyser le trafic au delà du niveau 4 (Spamd pour le SMTP ou ftp-proxy pour le FTP) sont développés dans l'espace utilisateur. Cette implémentation offre un niveau de sécurité supplémentaire par rapport à iptables qui fait tout dans l'espace noyau.

# 3 Architecture de FIELDS

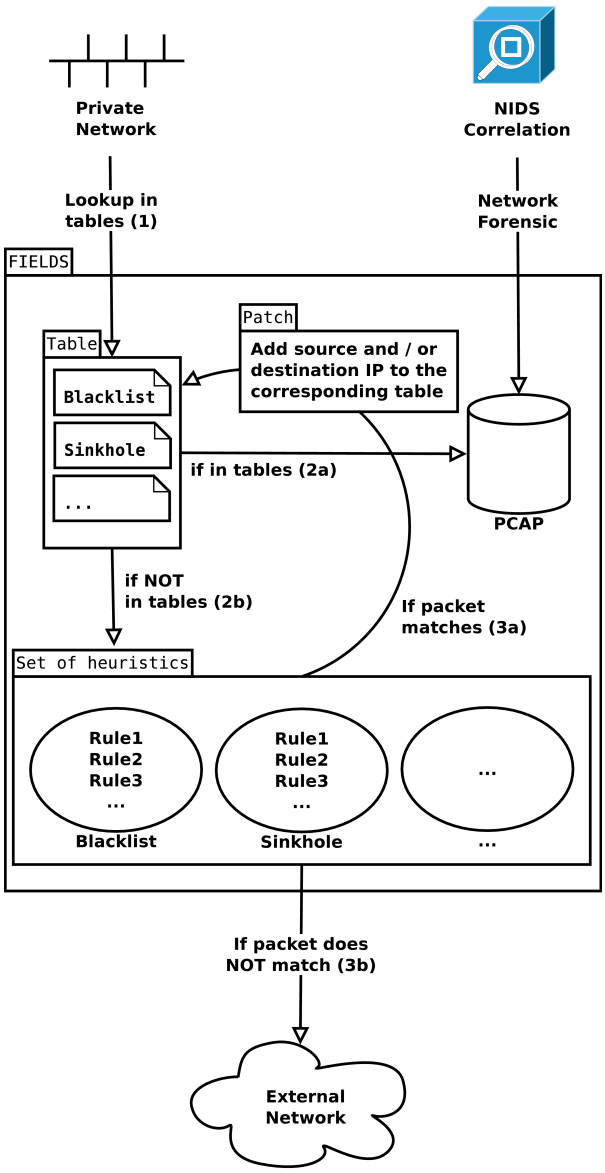


*Figure 2 - Architecture de FIELDS*

L'idée derrière FIELDS est d'utiliser la base PF pour modéliser les différentes heuristiques et de placer sous surveillance les machines ayant des flux réseaux suspects. Ce placement sous surveillance signifie que tous les paquets provenant ou à destination de ces machines suspectes sont enregistrés.

Le développement étend la grammaire de PF. Lorsque un paquet est évalué, trois actions peuvent être définies par la règles : « pass », « block » ou « log ». Le patch ajoute une option « add ». Cette option prend au moins un des deux arguments « ipsrc » et / ou « ipdst » qui pointent respectivement sur l'adresse source ou la destination de la règle. Chacun de ces deux arguments doit être suivi d'un nom de table dans laquelle l'adresse concernée est ajoutée. Petit exemple :

```
pass in on em0 from 192.168.0.0/16 to any
add ipdst <tableA>
```

Cette règle ajoute toutes les adresses de machines du réseau 192.168.0.0/16 traversant la sonde vers un réseau distant dans la table « tableA ».

La figure 2 détail le fonctionnement de FIELDS. Le paquet arrive du réseau privé (1). Il passe par les tables générées par les différentes heuristiques. Si le paquet coïncide avec une de ces règles (2a) alors il est logué au format PCAP pour exploitation (analyse en détail via un NIDS ou stockage à plusieurs niveaux de rétention avec Argus). Si le paquet ne satisfait aucune des tables, alors il passe par le jeu de règles modélisant les différents heuristiques (2b). Si le paquet suit une des heuristiques (3a) il est alors ajoutée dans la table correspondante et logué au format PCAP. Sinon il sort directement (3b).

## 3.1 Modification de PF (partie noyau)

Il faut commencer par créer une structure add dans le fichier pfvar.h. Cette structure contient un booléen pour indiquer sa présence ou pas dans la règle. Viennent ensuite deux structures pour la source et la destination. Chacune de ces structures contient une variable de type entier pour indiquer si il faut ou pas ajouter la source et / ou la destination, une chaîne de caractère contenant le nom de la table et un pointeur sur l'emplacement de cette dernière en mémoire. Cette structure add est ensuite ajoutée à la structure « pf_rule » contenant tous les paramètres de la règle.

```
struct add {
    u_int8_t check_add ;
    struct {
          u_int8_t check_src ;
          char src_tblname [ PF_TABLE_NAME_SIZE ] ;
          struct pfr_ktable *src_tbl ;
    } src ;
    struct {
          u_int8_t check_dst ;
```

```
            char dst_tblname [ PF_TABLE_NAME_SIZE ] ;
            struct pfr_ktable *dst_tbl ;
    } dst ;
} add ;
```

Une fonction « pf_save_addrs » a également été ajoutée. Celle ci extrait les adresses source et destination de la fonction « pf_pdesc » pour éventuellement les ajouter dans les tables spécifiées par la structure « add » contenue dans la structure « pf_rule » passée en paramètre de la fonction.

```
void pf_save_addrs (struct pf_rule ** rule , struct pf_pdesc *pd)
```

Cette fonction est appelée dans « pf_test_tcp() », « pf_test_udp() », « pf_test_icmp() » et « pf_test_other() ». Ces fonctions sont appelées lors du passage du premier paquet de la connexion. Pour les suivants PF maintient une table d'état des connexions en cours. Voici un extrait de la fonction « pf_test() » qui gère la correspondance règle / paquet pour TCP :

```
switch ( packet_protocol )
case TCP : {
    pf_test_state_tcp () ;
    if ( existing state )
            update state
    else
            pf_test_tcp ()
    break ; }
```

La fonction « pf_test_tcp() » est appelée à la première superposition positive règle / paquet. C'est ici que l'instrumentation a lieu :

```
if ( add option ) {
    pf_save_addrs (&r , pd) ; }
```

### 3.2 Modification de pfctl (espace utilisateur)

La commande pfctl interagit sur PF depuis l'espace utilisateur. L'intégration de FIELDS s'est résumée à modifier la grammaire Lex / Yacc pour intégrer le nouveau mot clé « add ». La fonction print_rule() a aussi du être modifiée pour prendre en compte l'affichage de la nouvelle option dans le jeu de règles.

L'intégralité du code de FIELDS est consultable ici : http://fields-flow-intrusion-extrusion-largescale-detection-system.googlecode.com/svn/add.patch

## 4 Heuristiques

Nous allons présenter deux heuristiques (d'autres ont étés modélisées [9]) : la blacklist et le sinkhole IP. Ces

heuristiques sont orientées extrusion. Celles ci peuvent être considérées comme une politique NSM (Network Security Monitoring) dans la mesure ou le but est de collecter du trafic suspect.

## 4.1 Blacklist

Il est possible de récupérer des liste d'IP de machines connues comme relais de SPAM, tête de réseau de botnet etc. L'idée est de mettre ces IP dans une liste et de placer toutes les machines de notre réseau ayant des interactions avec des machines de cette liste dans une table PF des machines suspectes afin d'en capturer le trafic. Au niveau heuristiques, le jeu de règles est le suivant :

```
table <blacklist>
table <compromised_hosts>

pass in log on $if from 192.168.40.0/24 to <blacklist> no state add ipsrc
<compromised_hosts>
pass in log on $if from <compromised_hosts> to any no state
pass in log on $if from any to <compromised_hosts> no state
```

La première règle trace les communications émanant du réseau interne 192.168.40.0/24 vers les machines de la table « blacklist ». Si un paquet coricide avec cette règle, son IP source est ajoutée à la table « compromised_hosts ». Les deux règles suivantes capturent le trafic provenant et à destination des machines de la table « compromised_hosts ».

## 4.2 Sinkhole IP

Comme il est indiqué dans la section 2.1, un sinkhole IP traditionnel ne récupère que les paquets à destination du sinkhole et nécessite une instrumentation des équipements de routage.

```
pass in log on { em0 em1 } from $FEDE to $sinkhole no state add ip src
<univ_to_sinkhole>
pass in quick log on { em0 em1 } from <univ_to_sinkhole> to any no state
pass in quick log on { em0 em1 } from any to < univ_to_sinkhole> no state
pass quick on { em0 em1 } from $FEDE to $FEDE no state
```

Il faut rappeler que dans PF c'est la dernière règle du jeu qui coïncide qui l'emporte sauf si le mot « quick » est présent dans la règle, dans ce cas l'évaluation s’arrête si le paquet coïncide avec cette règle. La variable « $FEDE » contient toutes les plages réseaux utilisées sur le campus, la variable « $sinkhole » contient la liste des plages RFC 1918 et la table « univ_to_sinkhole » contient toutes les adresses des machines du réseau ayant envoyé des données vers un réseau privé non utilisé. La première règle match sur un paquet envoyé vers un réseau privé pour ajouter l'IP source à la table « univ_to_sinkhole » et l'évaluation continue. Les deux règles suivantes journalise les paquets provenant ou à destination d'une machine dans la table « univ_to_sinkhole » et l'évaluation s'arrête. La dernière règle modélise une machine du campus parlant à une autre machine du campus et sort. Cette dernière règle évite que toutes les machines parlant à des réseaux privés existants sur le campus ne soient ajoutées à la table « univ_to_sinkhole » par la première règle.

# 5 Expérimentations

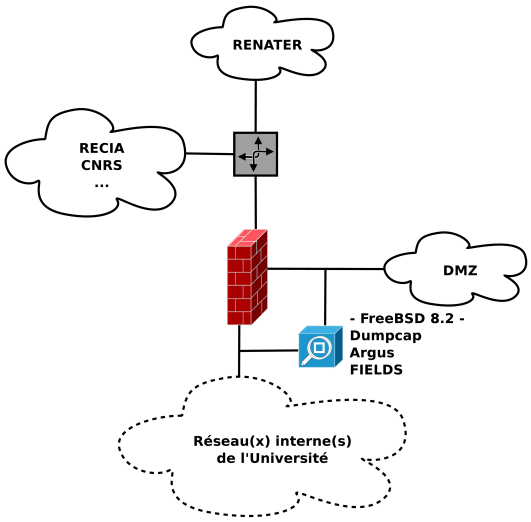


*Figure 3 - Topologie expérimentale*

La phase d'expérimentation a duré deux mois sur le campus de l'université d'Orléans. Les liens réseaux internes / pare-feu et DMZ / pare-feu ont été équipés d'un TAP réseau pour répliquer le trafic et l'inspecter de manière non intrusive. FIELDS a été installé sur une machine dotée d'un processeur Intel Xeon 4 cœurs avec 8GB de RAM. Le total du trafic à traiter est d'environ 1,5 TB journalier.

## 5.1 Démarche expérimentale

L'université d'Orléans est structuré en DSI. Il y a une équipe chargée de gérer les services centraux. Cette équipe travaille en relation avec un réseaux de correspondants qui gèrent localement chaque entité. Lorsqu'une machine présentait un trafic suspect, le correspondant chargé de l'entité concernée était avisé. Selon l'heuristique, il réalisait quelques manipulations pour essayer de confirmer ou d'infirmer la compromission de la machine. Il en est ressorti trois résultats possibles : compromission, indéterminé mais aussi des erreurs de configuration (machine pointant sur un DNS qui n'existe plus, accès scriptés sur des partages supprimés, serveurs ayant changé d'adresses etc.). Il arrivait aussi que par manque de temps, un correspondant ne puisse pas analyser la machine et la formate.

## 5.2 Résultats

Les résultats sont présentés dans le tableau suivant :

| Heuristiques | Nombre d'IP | Infecté | Configuration | Non confirmé | Formaté |
|---|---|---|---|---|---|
| Blacklist | 10 | 90 % | 0 % | 10 % | 0 % |
| Bruteforce | 0 | 0 % | 0 % | 0 % | 0 % |
| Sinkhole IP | 163 | 22 % | 11 % | 32 % | 35 % |
| Sinkhole DNS | 45 | 82 % | 0 % | 18 % | 0 % |
| Sinkhole Windows | 44 | 64 % | 15 % | 21 % | 0 % |

Le nombre d'IP donne la quantité de machines ajoutées dans les tables des différentes heuristiques. L'état « infecté » indique qu'après analyse antivirale de la machine ou analyse du PCAP la concernant avec le NIDS Snort le résultat était positif. L'état configuration indique le nombre de machines ayant un défaut de configuration détecté par FIELDS. « Non confirmé » indique que des investigations infructueuses ont été réalisées sur la machine. L'état formaté veut dire que la machine a été réinitialisée sans autre forme de procès.

# 6 Conclusion

FIELDS est donc un outil non intrusif pour réaliser du pré-traitement de trafic de manière non intrusive. Une simple recopie de port sur un routeur ou l'achat de boîtiers TAP réseaux suffit à le mettre en œuvre. L'expressivité du langage de PF combinée aux possibilités de proxy dans l'espace utilisateur font que l'on peut réaliser des choses assez raffinées au niveau analyse de flux.

# Bibiographie

[1] M. Ranum, A. Lambeth, and E. Wall, "Implementing a generalized tool for network monitoring," in In Proc. 11th Systems Administration Conference (LISA), 1997.

[2] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomaly detection using software defined networking,” in Proceedings of the 14th international conference on Recent Advances in Intrusion Detection, ser. RAID’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 161–180.

[3] R. Sommer and V. Paxson, “Enhancing byte-level network intrusion detection signatures with context,” in In Proc, 10th Conference On Computer And Communications Security. ACM, 2003, pp. 262–271.

[4] A. Caglayan, M. Toothaker, D. Drapaeau, D. Burke, and G. Eaton, “Behavioral analysis of fast flux service networks,” in Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies, ser CSIIRW ’09. New York, NY, USA: ACM, 2009, pp. 48:1–48:4.

[5] H.-G. Lee, S.-S. Choi, and Y.-S. L. H.-S. Park, “Enhanced sinkhole system by improving post-processing mechanism,” Future Generation Information Technology, pp. 469–480, 2010.

[6] B. Greene and D. McPherson, “Isp security: Deploying and using sinkholes,” NANOG talk, http://www.nanog.org/mtg0306/sink.html, 2003.

[7] R. Bejtlich, The Tao of network security monitoring: beyond intrusion detection. Addison-Wesley Professional, 2004.

[8] R. Bejtlich, Extrusion detection: security monitoring for internal intrusions. Addison-Wesley Professional, 2005.

[9] Briffaut, J., Grenèche, N., Narvor, Q., & Toinard, C. (2012). FIELDS: Flow Intrusion Extrusion Largescale Detection System. In The Sixth International Conference on Emerging Security Information, Systems and Technologies.