# FIELDS: Flow Intrusion Extrusion Largescale Detection System

Nicolas Grenèche
*Université Paris 13 — PRES Sorbonne Paris Cité*
*LIPN UMR CNRS 7030 / DSI*
*99 Avenue Jean-Baptiste Clément*
*93430 Villetaneuse (France)*
*Email: nicolas.greneche@univ-paris13.fr*

Quentin Narvor, Jérémy Briffaut, Christian Toinard
*ENSI de Bourges, LIFO — EA 4022*
*88 Bld Lahitolle*
*F-18020 Bourges Cedex, France*
*Email: {quentin.narvor,jeremy.briffaut,*
*christian.toinard}@ensi-bourges.fr*

***Abstract*—This paper presents an advanced pre-processing, called FIELDS, for analyzing the network traffic based on flow assessments. FIELDS is an extensible Network Security Monitoring that supports 1) advanced traffic pre-processing, 2) forensics and 3) existing Network Extrusion/Intrusion Detection Systems. FIELDS has been experimented during two months using a large real network thanks to its non intrusive nature. The results show the efficiency of different heuristics for pre-processing the traffic relevant of an intrusion. FIELDS provides an unified and efficient tool for pre-processing the network traffic and detecting/controlling the potential internal/external intruders. FIELDS solves the problem of scalability for the monitoring of the security of large networks. It can be easily extended to integrate other heuristics and correlate the different analysis.**

***Keywords-Network flows; Intrusion Detection; Network Forensic; Network Capture; Attack Prevention***

## I. INTRODUCTION

Network Security Monitoring is associated with different methods such as capturing the network traffic, analyzing the traffic, logging the traffic, forensics and network intrusion detection (NIDS). Classical NIDS has been studied since a long time [1]. They generally compute the ingoing and outgoing network traffic for detecting the intrusions. Studies mainly address how to learn [2] or modelize [3] a network threat. The major drawback is how to analyze the whole traffic in a scalable manner. FIELDS proposes a generic method for solving the problem of scalability. It provides a pre-processing of the outgoing traffic (pre-extrusion) as a mean to suspect the compromising of the local hosts (intrusion). Using the suspected list of hosts, FIELDS enables a safe Network Extrusion Detection System (NEDS) by reusing existing security tools (e.g. NIDS, anti-virus, anti-malware, HIDS, HIPS, ...) to measure the compromising. FIELDS provides an extensible and non-intrusive approach for 1) detecting potential intruders/victims based on efficient heuristics, 2) collecting the traffic of those eventually malicious hosts and 3) analyzing the corresponding flows and 3) reusing an existing packet filter while guaranteeing a low overhead. Despite FIELDS extrudes the outgoing traffic from the internal malicious hosts, it can provide the external malicious hosts. Thus, FIELDS can also improve the detection/prevention of intrusions coming from external hosts (pre-intrusion). Globally, FIELDS reduces the overhead of a NIDS/NEDS approach by permitting a consistent analysis only for a subset of the whole traffic associated with the internal/external suspected hosts. The second section describes the state of art related to heuristics for pre-processing the outgoing network traffic and to the network capture. A third section describes our major objectives for pre-processing the outgoing traffic, analyzing the suspected hosts and controlling/measuring the suspected hosts. The fourth section describes our solution. A fifth section describes our experimentation. Finally, the paper describes the perspectives and summarizes the FIELDS approach.

## II. STATE OF THE ART

Classical NIDS approaches such as [2], [3] need pre-processing in order to limit the overhead and reduce the number of false positive alerts. Thus, our state of art considers only the pre-processing problem in order to have 1) an updated list of the suspected hosts and 2) an exhaustive capture of the abnormal traffic coming from those hosts. This section shows which heuristics enable to compute a list of suspects based on extrusion, i.e. abnormal outgoing traffic from internal hosts that is a consequence of a successful intrusion. Then, the section goes on with the methods available for capturing the traffic of the suspected hosts. Finally, it describes recent works related to extrusion.

*IP blacklist:* This is the most simple heuristic. Despite it's simplicity, this is currently massively used to detect malwares that contain hardcoded addresses or names. For malwares associated with a more sophisticated infrastructure containing dynamic hosts e.g. associated with fast flux based methods [4], the IP blacklist approach becomes inefficient. IP Blacklist can be used for both intrusion and extrusion monitoring i.e. external and internal malicious hosts.

*DNS domain blacklist:* This blacklist maintains a list of malicious DNS domain names. This technique is extrusion oriented because only the DNS requests from internal hosts are audited. The DNS resolver of an infected host can also responds with an unused IP address belonging to a sinkhole.

 ISBN: 978-1-61208-209-7 

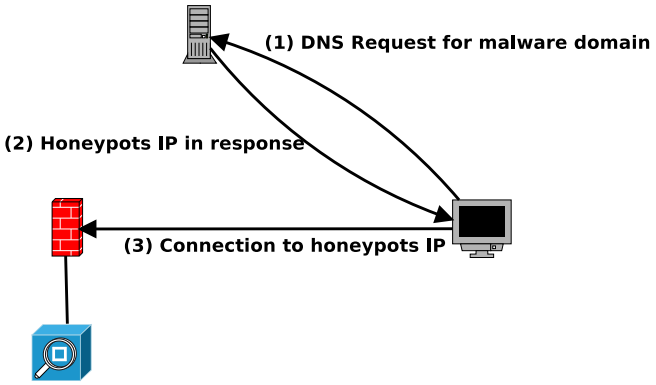


Figure 1. DNS name blacklist

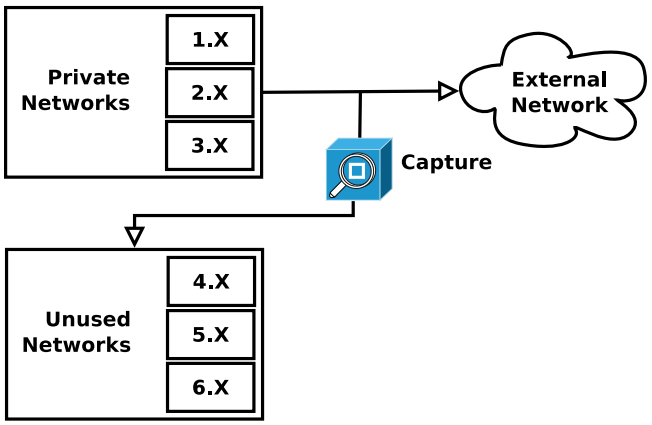


Figure 2. IP Sinkhole for 192.168.X.X/24

In this case, as shown in Figure 1, a honeypot can handle the connections of the malwares [5] to this address.

*IP Sinkhole:* ISP sinkholes are used at BGP level by Internet providers to redirect and log attacks against customers [6]. IP sinkholes are used by network administrators to monitor a traffic that tries to reach unused private networks [7], [8]. Private range of addresses are defined by the RFC1918. Addresses of the RFC1918, that are not used by the organization, are in the sinkhole. An implementation using a router is given in Figure 2. There are three real networks (192.168.1.0/24, 192.168.2.0/24 and 192.168.3.0/24). Routing tables are set up to route real private networks first. Then, RFC1918 networks are routed. Finally, a last route authorizes the Internet access. As a consequence, if a packet tries to reach a non existent (but belonging to RFC1918) network, it is captured by routing equipments. However, this solution is intrusive because it needs to modify the routing tables.

*Network capture:* As stated in [7], [8], [9] four different kinds of information can be stored for Network Security Monitoring (NSM): global statistics (how many data sent/received to/from a given network), flows (who talked to who and how), IDS alerts and comprehensive captures. For large organization, a comprehensive capture of each network flow is impossible but such level of detail is interesting in terms of network forensics. Common network sensors can be configured to perform a comprehensive capture (tcpdump) or only a capture of the flows (netflow, sflow or argus collector). Currently, there is no hybrid collector available in the literature. An hybrid collector would be able to switch from one capture mode to another one to zoom on supposed malicious flows.

*Recent extrusion works:* Solutions such as [10] proposes to install viruses and malwares for analyzing their behaviour. Others [11] reuses snort for analyzing outgoing traffic. However, those approaches consider analysis of viruses and do not solve the problem of scalability for analyzing the whole outgoing traffic.

## III. MOTIVATIONS

The first objective is to suspect a host using a network extrusion. It is interesting since extrusion does not require to analyze the ingoing traffic reducing thus the overhead of the analysis. Moreover, it allows to detect threats coming also from usb or other removable resources and the hosts do not have to be monitored. The second objective consists in reusing simple tools efficient for capturing the traffic. It is a low cost approach since tools such as BSD Packet Filter are free and strongly supported. Third, our approach must support different types of heuristic for computing the list of the suspected hosts. Indeed, the IP/DNS blacklist and sinkhole can miss some infected hosts. Thus, the list of the malicious hosts can be computed using the the IP/DNS blacklist and sinkhole approaches and improved with other methods such as CERT notifications, OS fingerprinting or IDS alerts. Fourth, the approach must reuse existing security tools (e.g. anti-virus, anti- malware, HIDS, HIPS, NIDS, ...) to measure or recover the infected nodes. Fifth, the solution must solve the scalability problem by analyzing part of a network traffic that is limited to to the suspected nodes. Sixth, the network analysis must be non-intrusive and require only a limited modification of the capturing tool. Seventh, the monitoring solution must support not only internal hosts but also external hosts. The advantage is a common and extensible approach to manage and reduce the overhead of NEDS/NIDS tools.

## IV. THE FIELDS APPROACH

FIELDS is a software that uses the BSD Packet Filter (PF) [12] tables and PCAP logging capabilities. It requires only a very limited modification of PF while supporting a large range of heuristics. In PF, a table provides a list of IP addresses matching a destination or a source. A table referred to as `<suspicious>` can be declared to store potentially infected hosts. The lookup operation on the table is very efficient. The logging infrastructure of PF enables to

 ISBN: 978-1-61208-209-7 

capture network packets that match a given rule in the PCAP format. For example, the following rule enables to capture all the traffic originating from the IP addresses available in the <suspicious> list:

```
pass in log on em0 from <suspicious> to any
```

The idea of FIELDS is to express malicious flows through a dedicated policy formalizing different heuristics. This way, FIELDS highlights malicious flows. When such a flow is detected, FIELDS switches into a comprehensive capture of the traffic. FIELDS can be considered as a kind of NEDS/NIDS pre-processor reducing the data size that has to be analyzed by existing NEDS/NIDS. In that sense, it is not concurrent of existing approaches but eases their management in a production environment in order to solve the scalability problem while reusing off-the-shelf NEDS/NIDS tools.

## V. PACKET FILTER PATCH

FIELDS uses a modified Packet Filter. The proposed patch extends the capabilities of the PF rules. When a packet matches a rule, PF performs some actions: `pass`, `block` or `log`. Our patch adds a fourth action `add` which performs the addition of a source and/or destination address in a table. The Figure 3 describes our patch integration into PF. When a packet reaches the PF firewall, a lookup in the tables is preformed (step 1). A dedicated table can be declared for each malicious flows heuristic: a table <blacklist> to store the IP addresses that try to reach the hosts of the blacklist, a table <sinkhole> to store the addresses talking to the sinkhole addresses, etc. If the destination or source (depending of the meaning of the matching rule) is present in a table, the packet is logged (step 2a). If the corresponding address is not in a table, the packet is submitted to a set of filtering rules modelizing a set of malicious flows heuristics (step 2b). If the packet matches with a rule, its source and / or destination IP is added to the corresponding table (step 3a). For example, if a packet matches a rule describing a sinkhole heuristic, its source IP is added to the <sinkhole> table. Thus, any packet involving this IP address will be logged (step 2a). These logs are analyzed afterwards by different IDS tools (such as snort) in order to identify malwares. FIELDS extends the grammar of the PF rules. Our patch is about 300 lines of C, covering kernel code (100 lines) and userland `pfctl` command (200 lines). This patch is not intrusive since it does not modify the sensible parts of the kernel.

### A. Extension of the PF grammar

A new option has been added in the rules syntax: the option `add`. This option must be used after all the usual options of pf. `add` have to be followed by at least one of these 2 options:

- `ipsrc <src_tblname>`: add source IP address of the matched packet to the table "`src_tblname`" ;

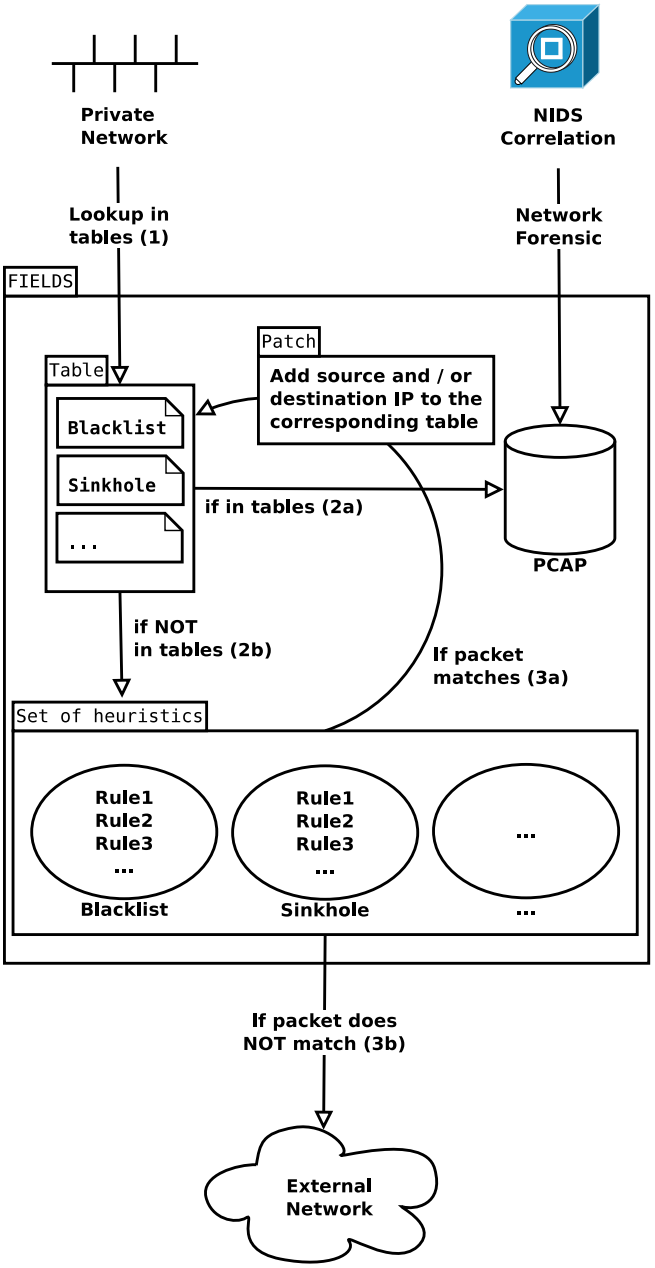


Figure 3. FIELDS architecture

- `ipdst <dst_tblname>`: add destination IP adress of the matched packet to the table "`dst_tblname`".

Let's give examples of such a rule:

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

In this rule, the destination addresses of all the packets coming on the network interface em0 from the subnet 192.168.0.0/16 will be added to the table named `tableA`.

 ISBN: 978-1-61208-209-7 

```
pass in on em0 from 192.168.1.0/24 to 192.168.2.0/24 add
    ipsrc <tableA> ipdst <tableB>
```

In this rule, the source addresses of all the packets coming on em0 from the subnet 192.168.1.0/24 to the subnet 192.168.2.0/24 will be added to the table named tableA. In the same way, destination addresses will be added to the second table named tableB.

### B. Modifications on PF

A new structure named add has been added in the file pfvar.h. This structure stores data required by our new options:

```
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;
```

The existence of the add keyword in a filtering rule is flagged with the check_add member. This keyword wait for source or destination IP addresses associated with the tables required to store those addresses. Those IP addresses can be assigned to different tables, i.e. the source address can be associated with a table and the destination address with another table. Thus, this structure allocates two sub structures src and dst. Each sub structure contains a variable check_[src|dst] (that flags the existence of source and / or destination IP address), [src|dst]_tablname (name of the table used to store the IP address) and [src|dst]_tbl (a pointer on the table data).

In the file pf.c, a function pf_save_addrs has been added:

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

This function extracts the source and the destination addresses of the packet from the pf_pdesc structure and adds them into the tables specified on the pf_rule structure.

This function is called four times respectively in pf_test_tcp(), pf_test_udp(), pf_test_icmp() and pf_test_other() for covering the different levels of the IP protocols. Those four functions are called for the first packet of a connection. For the following packets, PF maintains a state for the corresponding flow. Here is an extract of pf_test() (pf_test6() for IPv6) function which handles the packet matching for TCP:

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

The function pf_test_tcp() is thus called for the first packet of a TCP connection (line 7). This is where the instrumentation, i.e. the saving of the addresses, must be done. In pf_test_tcp():

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

### C. Modifications on pfctl

Pfctl is the userland tool to interact with PF. Its main function is to enable, disable and check a ruleset stored in a configuration file pf.conf. It also permits to interact with some PF objects on the fly (for example adding an IP address to a table, list IP addresses of a table, load an anchor etc.).

PF makes use of Lex and Yacc to check a ruleset. The Yacc grammar has been modified to take the add keyword and its options (ipsrc and ipdst) in account.

In the file pfctl_parser.c, the add keyword has to be taken into account in the function print_rule() that displays the current ruleset using the pfctl command:

```
if (r->add.check_add != 0) {
   if (r->add.src.check_src != 0 && r->add.dst.check_dst
       == 0)
      printf("add ipsrc <%s>", r->add.src.src_tblname);
   if (r->add.src.check_src == 0 && r->add.dst.check_dst
       != 0)
      printf("add ipdst <%s>", r->add.dst.dst_tblname);
   if (r->add.src.check\_src != 0 && r->add.dst.check_dst
       != 0)
      printf("add ipsrc <%s> ipdst <%s>", r->add.src.
          src_tblname, r->add.dst.dst_tblname);
}
```

Line 1, a check is performed on r (current rule) to see if the add keyword is present on the rule through the member check_add. If present, a check to see if the source IP (line 2), destination IP (line 4) or both (line 6) have been specified for logging.

## VI. FIELDS: NEW NSM HEURISTICS

### A. IP Blacklist

Some websites, such as spamhaus, uce-protect and so on, provide blacklists filled in by addresses that have been judged compromised: botnet, spam, etc... These lists can be used to block an incoming connection to the internal network hosts. Connections incoming from these addresses are blocked by the firewall but connections made from internal hosts to these addresses are not controlled. Logging these connections is interesting because it means that the host has a suspect behavior.

The FIELDS controller can detect these connections and then automatically launch a comprehensive capture of these

internal hosts for a further analysis. Since those internal hosts have a suspect behavior, a deep analysis of their traffics might tell us if they are infected, and eventually, with which malware.

A simple bash script has been made to automatically download updated blacklists and load them into FIELDS. This is done without reloading the entire policy of FIELDS thanks to `pfctl`.

As pf can handle lots of distinct tables, the system is very modular. The comprehensive capture can be optional: the first suspect packet is always automatically logged, and the address added into the desired table. This table just can be used to know which host has match a given rule. Example:

```
table <blacklist>
table <compromised_hosts>
table <internals_hosts>

pass in log on $if from <internals_hosts> 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
```

These rules allow `pf` to launch a comprehensive capture on any internals hosts of our network which try to connect to a host from the blacklist. There are two stages is the FIELDS process:

- Addition of the suspected host that talks to a blacklisted address to the table `<compromised_hosts>` (line 5) ;
- Comprehensive capture of each packet from (line 7) and destined to (line 8) the suspects stored in `<compromised_hosts>`.

### B. IP Sinkhole

FIELDS can modelize an IP sinkhole. The collection of internal subnets is `$FEDE`, RFC1918 addresses are stored in `$sinkhole`. The `em0` and `em1` interfaces are linked to the Network TAP. FIELDS is listening on them.

```
pass in log on { em0 em1 } from $FEDE to $sinkhole no
     state add ipsrc <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
```

The traffic from the internal network to the sinkhole is logged and the sources are added to `<univ_to_sinkhole>` (line 1). Traffic from (line 2) or destined to (line 3) hosts belonging to `<univ_to_sinkhole>` is dumped and the packet evaluation ends ("quick" keyword). Packets from internal to another internal network pass without any log (line 4).

### C. DNS Sinkhole

FIELDS can detect when an internal host attempts to connect to a blacklisted address, as shown in Figure 4. But,

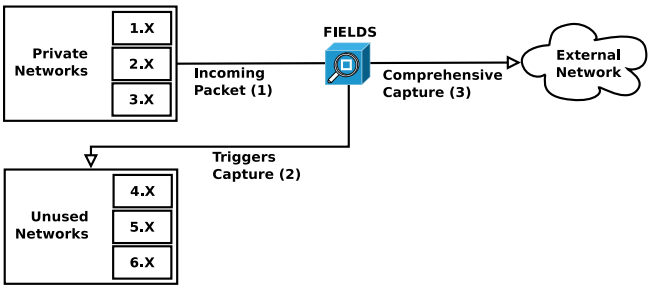


Figure 4. FIELDS IP / DNS Sinkhole for 192.168.X.X/24

the connections can be done with malware domains. To cover such connections a DNS Sinkhole is available. Thanks to the DNS Sinkhole, our DNS gives a false IP address as a response to a DNS request for all the domains which are considered as compromised. Our DNS Sinkhole lists the malware domains e.g. the domains provided by dedicated websites covering well-know malware domains. Thanks to our DNS Sinkhole, the host that will try to establish a communication with a suspected domain will fail, but it will generate an odd traffic which will be recognized by FIELDS. This false IP address has to be a private IP address (rfc 1918), and will be recognized as a part of the sinkhole. The rule concerning the DNS has been written after the rule of the sinkhole to be sure it takes the priority. Otherwise, we would not be able to see the difference between a host making a request for a blacklisted DNS and a host trying to contact a host in the sinkhole. A FIELDS policy enables to log all the hosts that attempt to connect to this false IP address. It probably means that this host is compromised by some malware and needs an antivirus/IDS analysis.

The DNS rules are close to the blacklist rules except we add our false IP address returned by the DNS to 1) the table `<compromised_hosts>` or 2) another table.

### D. Using the transport layer: Service sinkhole

Another variant of sinkhole is to rely on transport to determine which service on non-existant hosts is tried to be accessed. A Windows oriented sinkhole has been modelized to catch hosts that try to access Windows ports:

- 137: NetBIOS name service (UDP) NetBIOS-sn lookups for "`gethostbyaddr()`" function;
- 138: NetBIOS datagram (UDP) non connected messages exchange;
- 139: NetBIOS Session (TCP) Windows File and Printer Sharing;
- 445: CIFS (TCP) used for the Common Internet FileSystem resources sharing protocol;
- 1433: SQLServer (TCP) Microsoft SGBD.

 ISBN: 978-1-61208-209-7 

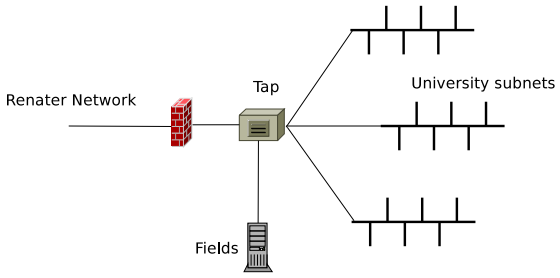


Figure 5. FIELDS integration topology

```
pass in log on { em0 em1 } proto udp from $FEDE to
     $sinkhole port { 137 138 139 } no state add ipsrc <
     to_winsinkhole>
pass in log on { em0 em1 } proto tcp from $FEDE to
     $sinkhole port { 445 1433 } no state add ipsrc <
     to_winsinkhole>
```

### E. Bruteforce

The extensibility of FIELDS easily enables to cover the bruteforce traffic attempting to discover the network resources. A FIELDS policy enables to support the network bruteforce. Thus, FIELDS captures all the traffic of a host that does the network scanning. Example:

```
pass in log on $if from <internals_hosts> to <
     internals_hosts> port { 22 445 } keep state (max-src-
     conn-rate 3/10, overload <bruteforce> flush global)
```

This rules add the address of any internal host that attempts a bruteforce attack on ssh and samba ports into the table `<bruteforce>`.

## VII. EXPERIMENTATIONS

As shown in Figure 5, FIELDS is placed behind a Network TAP. This device copy traffic passing through the firewall and the external router in a non-intrusive way. As a consequence, in case of performance lack / bug, the real network is not affected. FIELDS processes all the traffic between 1) the Internet and the University of Orleans, 2) each university pole, and 3) university poles and the DMZ. FIELDS contains only rules relative to the EDS/IDS pre-processing VI. Remember that the idea is not to supply a complete EDS/IDS but an intelligent pre-processor for exhaustively collecting only the traffic related to suspected hosts (and not only the suspected traffic).

As explained in V, FIELDS captures the traffic from lists of IPs stored in tables. When an IP is added to a table, a timestamp of the current date is included. If the IP is already stored in the table, PF only updates the record with the current date. In order to avoid having growing only tables, FIELDS also must be able to delete a record from a table. A deletion can be carried out manually when the host has been checked as safe by the security team. But, PF also provides automation for the deletion. For example, if there is no malicious activity involving this address for a given period of time, then FIELDS deletes the entry from the table. In practice, FIELDS uses the program `expiretable` (which is shipped with PF) for removing an address after a period of time (2 days on this study) without any illegal activity.

`Pflog` is used to log all the traffic of the selected hosts. In the proposed experimentation, these `pcap` files are processed with argus in order to find flow anomalies first, then with snort in order to detect more advanced attacks. Obviously, other IDS and network analysis are supported by FIELDS.

FIELDS can detect outgoing attacks: if an intruder already owns an host in our internal network, his ignorance of the network structure, his attempts to discover the network, will make his host detected by FIELDS (`sinkhole`). If he is detected, all the traffic of the suspected host will be logged, which will make the attack a lot easier to analyze. The flexibility of the tables of `pf` allow us to isolate potentially harmful traffic such as an `nmap` scan from an internal host. It's possible to isolate these IPs in a table, which will be read regularly. As FIELDS detects the traffic which is not supposed to happen, a host misconfiguration can also be detected.

### A. Detection results

This section presents the results of the FIELDS experimentation on a real network of about 30 000 users for a period of 2 months. The volumetric for such a network is about 1,5 TB per day of raw PCAP for the external link (roughly speaking, link from your network to the Internet). The analysis on each host is performed in two rounds: a scan with Snort on the captured PCAP and a local antiviral analysis (if Snort returns nothing). FIELDS has been installed on common hardware (Intel XEON processor / 4 cores and 8GB of RAM). Table I gives the number of IP caught by FIELDS. Blacklist has only 10 hosts caught. Those hosts are infected by `conficker` and tries to reach servers handled by Microsoft. No bruteforce attempts have been detected. It's a false negative because there are bruteforce attempts but they are too slow or they come from distributed sources. As a consequence, they do not raise an alert based on connection rate as exposed in VI-E. Majority of the hosts has been caught with sinkhole heuristics. Table I gives the proportion of:

- infected: malicious payload detected on captured PCAP or malicious programs detected on hosts ;
- misconfigured: bad settings at network level on host (WINS server, DNS settings, routing etc.) ;
- unconfirmed: traffic and antiviral analysis did not give any results ;
- unchecked: traffic analysis did not give any results and antiviral analysis has not been performed ;

 ISBN: 978-1-61208-209-7

Table I
DETECTION RESULTS

| Heuristic | Number of IP caught | Infected | Misconfigured | Unconfirmed | Unchecked |
|---|---|---|---|---|---|
| Blacklist | 10 | 90% | 0% | 10% | 0% |
| Bruteforce | 0 | 0% | 0% | 0% | 0% |
| IP sinkhole | 163 | 22% | 11% | 32% | 35% |
| DNS sinkhole | 45 | 82% | 0% | 18% | 0% |
| Windows sinkhole | 44 | 64% | 15% | 21% | 0% |

The only false positive for the blacklist is a GET request from the security team for testing purpose. IP sinkhole has numerous unchecked hosts because it is difficult to request local antiviral tests on those hosts. Moreover, the IP sinkhole is prone to false positive because there are a lot of situations that can lead to send packets to sinkhole destinations (essentially typos in command). The DNS sinkhole is very effective, false positive are performed by people accessing malicious websites (i.e. blacklisted DNS domain name) through their web browser. Snort did not detect malware on traffic of DNS sinkholed hosts, only antiviral analysis gave results.

### B. Network forensic

The amount of the whole PCAP captured is 46 Gb for a day. FIELDS enables to retain a lot more traffic than an exhaustive capture on the outgoing link (about 1,5 TB a day). This is great help for a forensic purpose when a request comes from upper IT security authority (such as RENATER CERT).

## VIII. CONCLUSION AND PERSPECTIVES

FIELDS provides some key features in network security monitoring:

- Modelization and improvements of malicious flows detection algorithms;
- Efficient pre-processor of flows based on a reduced improvement of a classical Packet Filter;
- Hybrid collector of network traffic helping network forensic process;
- Non intrusive monitoring ;

A large scale experimentation shows the efficiency of the approach and demonstrates the ability of FIELDS to take the best of existing security tools (antivirus, NIDS). Thus, the approach enables to confirm or to unconfirm the risk. The results encourage us to follow FIELDS in several directions. First, FIELDS eases the modelization of other heuristics. Second, it can help to prevent the propagation of the intrusions through firewall/network configurations since it provides relevant information about the network compromission. Finally, an efficient forensic could be proposed through correlations of different tools since FIELDS provides the malicious traffic.

## REFERENCES

[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/mtg-0306/sink. html*, 2003.

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

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

[9] C. Sanders, *Practical packet analysis*. No Starch Press, 2007.

[10] B. Sunny and K. Krishan, "An experimental analysis for malware detection using extrusions," ser. ICCCT. IEEE, 2011, pp. 474 – 478.

[11] T. Ankita and S. R. /, "Optimization of snort for extrusion and intrusion detection and prevention," *International Journal of Engineering Research and Applications, Vol. 2, Issue 3*, pp. 1768–1774, 2012.

[12] D. Hartmeier, "Design and performance of the openbsd stateful packet filter (pf)," [retrieved: 07, 2012], 2002.

 ISBN: 978-1-61208-209-7 