Massively distributed intrusions detection : goals, challenges and possible solutions.

SEC2 2015, Lille

Michaël Hauspie

CRIStAL, CNRS UMR 9189 – Équipe 2XS

Plan

Plan

Intrusion detection is hard in a cloud context

Cloud specific issues

- Complex and dynamic network architectures
- Sensible data

Attacks can avoid standard security solutions

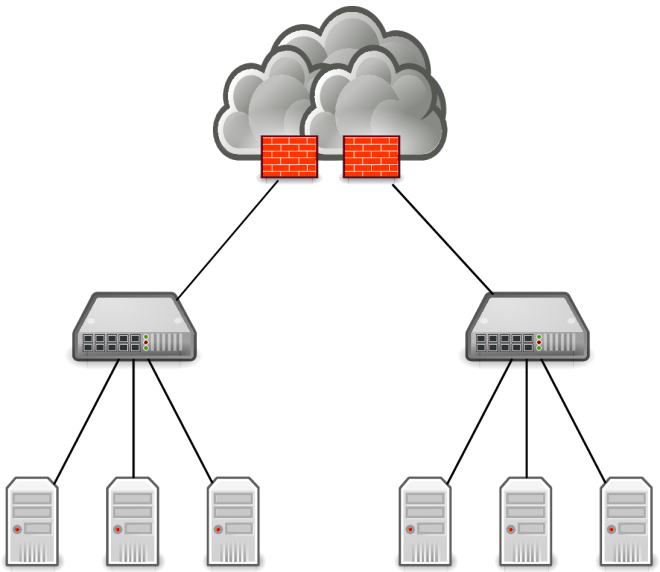
- by trying to lead the attack from inside the network [1]
- by spliting the attack using several hosts, network routes [4]

Bandwith and computing power is cheap to rent : Cloud As A Weapon [1]

D. Bryan and M. Anderson. *Cloud computing, a weapon of mass destruction*, DEFCON 2010

- Thunderclap
- Less than a few dollars to put a host down
- Instead of infecting host to create a botnet, just rent them !

Usual security solutions tends be located at the edge of the network



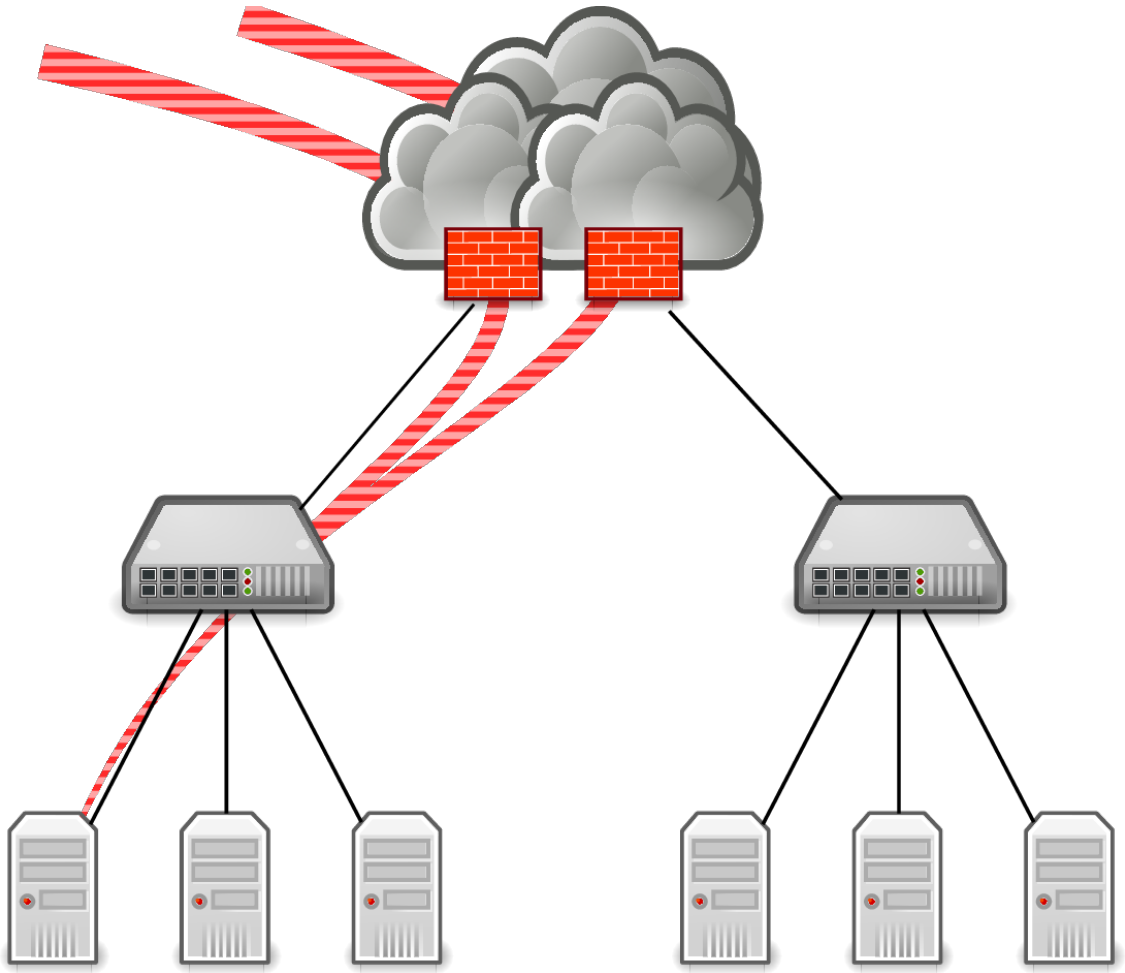
Firewalls

- usually located at the connection between data center and ISP
- filters network packet based on security rules

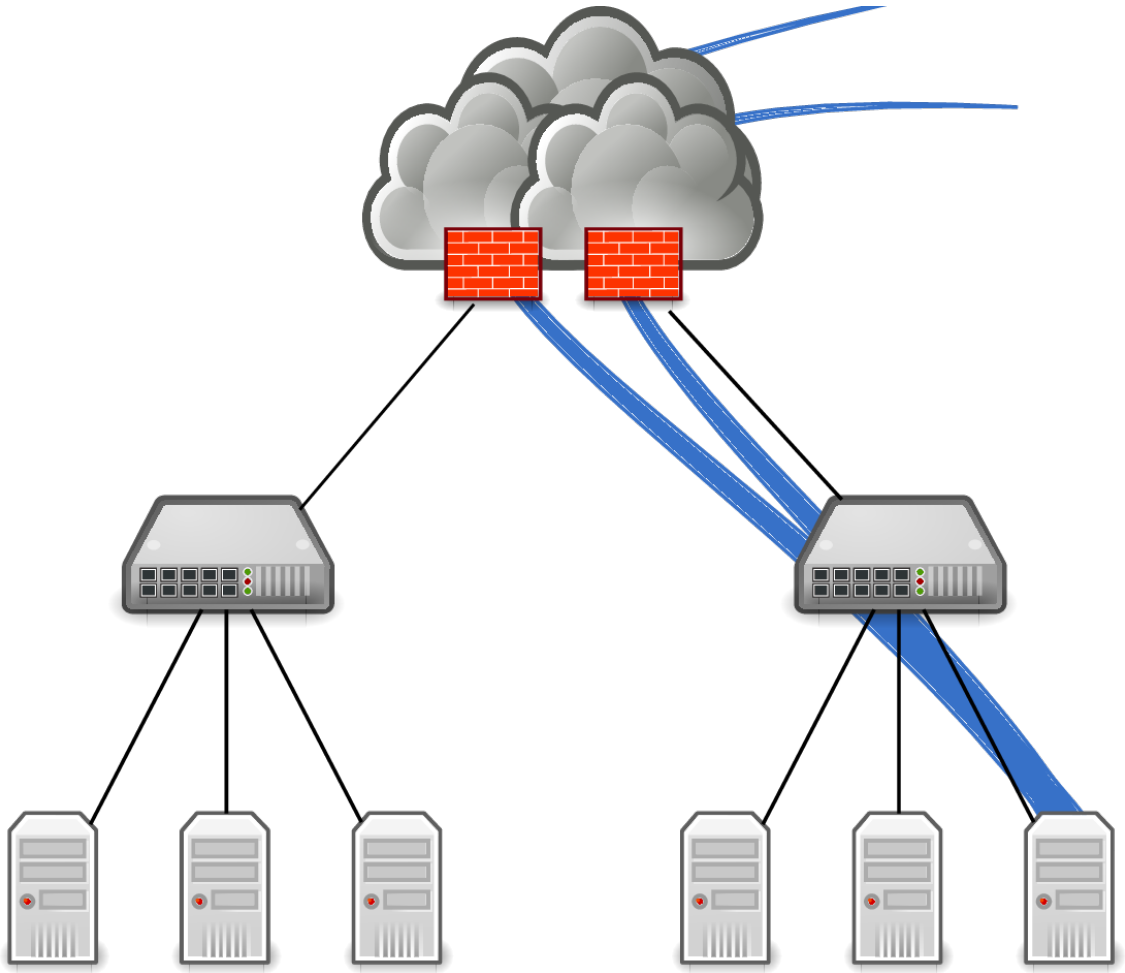
Intrusion Detection Systems (IDS)

- monitors the network (NIDS) or the operating system (HIDS)
- passive system : its goal is to raise alerts
- pattern matching or behavior analysis

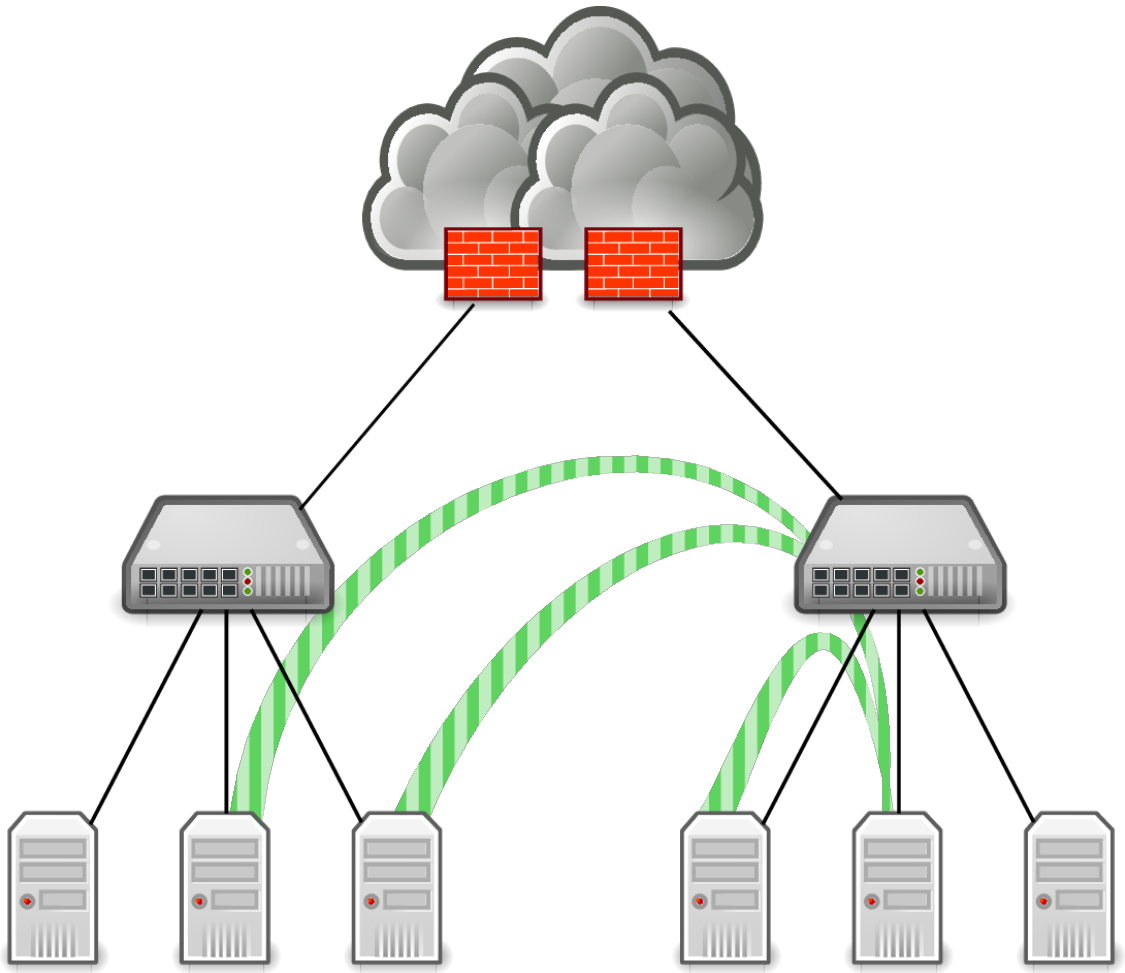
Attacks can come from outside



Attacks can be performed from inside



Attacks can stay inside



Plan

One solution may be distributed, collaborative IDS

Push IDS inside the infrastructure

- More probes means more information
- More information means better detection (or at least, may lead to)

One solution may be distributed, collaborative IDS

Push IDS inside the infrastructure

- More probes means more information
- More information means better detection (or at least, may lead to)

Why not almost everywhere ?

- Firewalls
- Switches
- Network cards/link
- Hypervisors

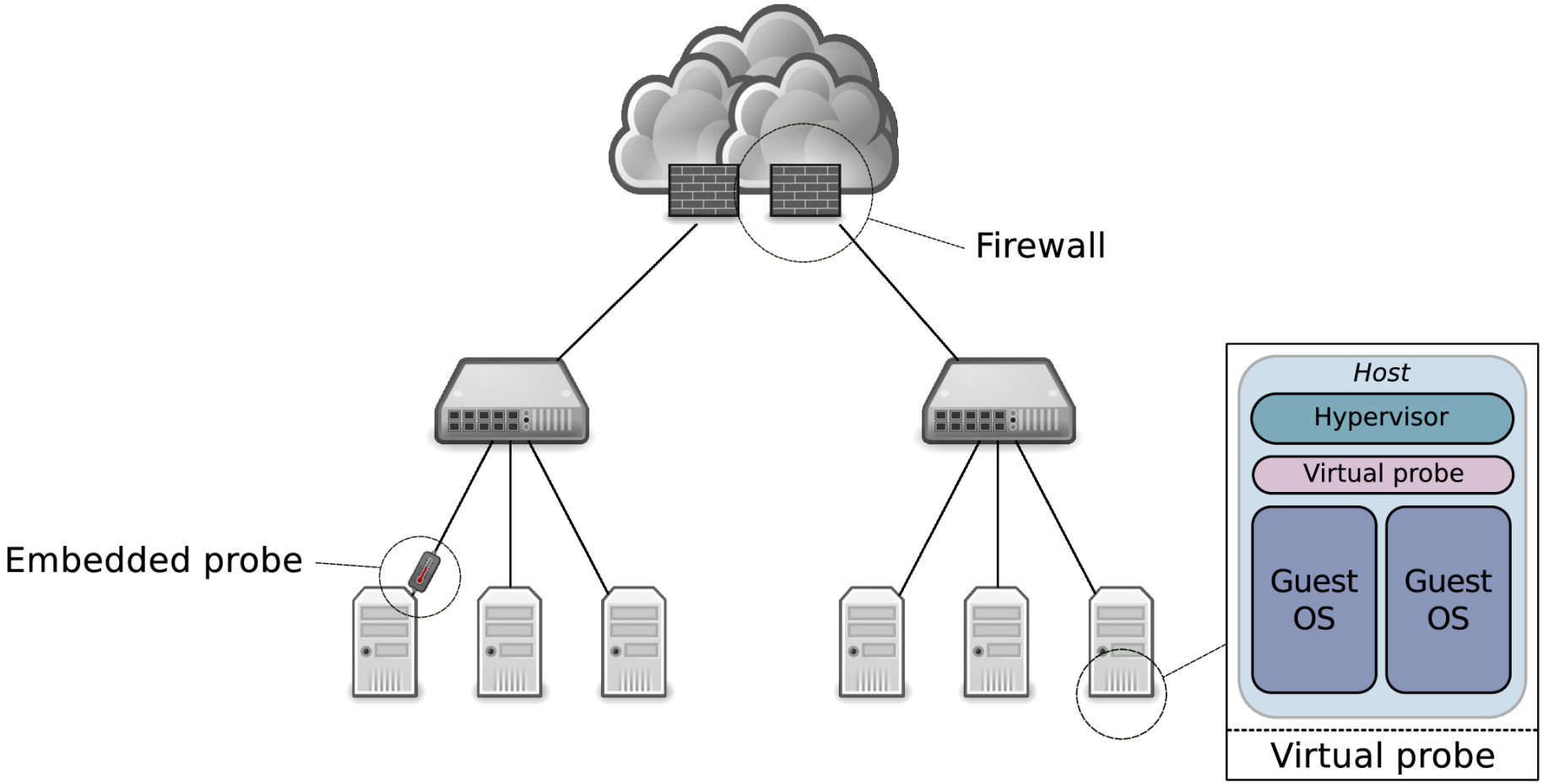
Plan

DISCUS is our proposal to deploy IDS everywhere [3]

Main ideas

- Put IDS probes as close to monitoring targets as possible
- Probes can be software or hardware
  - Embedded : cheap , but not very powerful, and hard to program
  - FPGA : very good power/cost ratio , but hard to create
  - Kernel or userspace software (snort, standalone software, kernel module) : can achieve high performance but need powerful hardware (high cost)

Let's put probes everywhere



Creating software for the probes is not that easy

Issues

- Heterogeneous targets $\rightarrow$ lots of expertise
- Collaboration is hard
- Deployment is hard

Creating software for the probes is not that easy

Issues

- Heterogeneous targets $\rightarrow$ lots of expertise
- Collaboration is hard
- Deployment is hard

Use of a Domain Specific Language (DSL)

- Focus on detection logic, not implementation details
- Use compile tools to handle heterogeneity and deployment

DSL focus on specific logic

- Not as expressive as generic purpose languages

DSL focus on specific logic

- Not as expressive as generic purpose languages
  - Limit development errors
  - Ensure strong properties on generated software
  - Allow better automatic optimisation

Event syntax

```
on my_event_name(args_list)
    [where condition]
    [...]
    action_list
    [...];
```

Actions

- Raise another event (now or later)
- Raise an alert
- Handle tables' structure

Filtering HTTP packets

```
on tcp_packet(..., int16 dst_port, ...)
    where dst_port == 80
    raise http_packet(...);
```

Tables : a structure to share data

- Distributed database of contextual data
- Table entries are aggregates of primary types
- Provides a way to collaborate

Declaration of a table

```
table tcp_connection {
    ipaddr src,dst;
    int16 p_src,p_dst;
    enum tcp_connection_state state;
    time last_pkt;
    (...)
};
```

Removing entries and purging tables

```
remove entry from tcp_connection when
    entry.state == TCP_CLOSED;

on purge tcp_connection select entry where
    entry.last_pkt + 3600 < now;
```

Using tables in events : updating last packet timestamp

```
on tcp_packet(ipaddr src, ipaddr dst, ...)

        update entry.last_pkt = now
```

Using tables in events : updating last packet timestamp

```
on tcp_packet(ipaddr src, ipaddr dst, ...)

    for first entry in tcp_connection with
            entry.src == src
            and entry.dst == dst
            and (...)

        update entry.last_pkt = now
```

Using tables in events : updating last packet timestamp

```
on tcp_packet(ipaddr src, ipaddr dst, ...)

    for first entry in tcp_connection with
            entry.src == src
            and entry.dst == dst
            and (...)

        update entry.last_pkt = now

    ifnone
        insert into tcp_connection {
            src = src;
            dst = dst;
            state = TCP_INIT;
            last_pkt = now;
            (...)
        };
```

Use case : detecting SYN Flood attacks

A basic DoS attack : SYN Flood [2]

- Opening a lot of TCP connections
- Initiating the handshake but not finishing it
- Memory congestion

Use case : detecting SYN Flood attacks

A basic DoS attack : SYN Flood [2]

- Opening a lot of TCP connections
- Initiating the handshake but not finishing it
- Memory congestion

Detection

- Count number of partially opened TCP connections per client
- Alert when the number reaches a fixed threshold (50)

Table

```
table dos_attempt {
    ipaddr src;
    int16 count;
    time last_attempt;
};


remove entry from dos_attempt
    when entry.count > 50;

on purge dos_attempt select entry
    where entry.last_attempt + 3600 < now;
```

Finding partially opened TCP connections

```
on tcp_packet(ipaddr src, ..., int9 flags, ...)
    where flags == tcp_flags.SYN
    raise check_dos_attempt(src, dst,
        src_port, dst_port) in 50 ms;
```

Finding partially opened TCP connections

```
on tcp_packet(ipaddr src, ..., int9 flags, ...)
    where flags == tcp_flags.SYN
    raise check_dos_attempt(src, dst,
        src_port, dst_port) in 50 ms;

on check_dos_attempt(ipaddr src, ..., int16 dst_port)
    for first entry in tcp_connections with
            entry.src == src and (...)
            and entry.connection_state != OPENED
        raise dos_attempt(src);
```

Dealing with DOS attempts

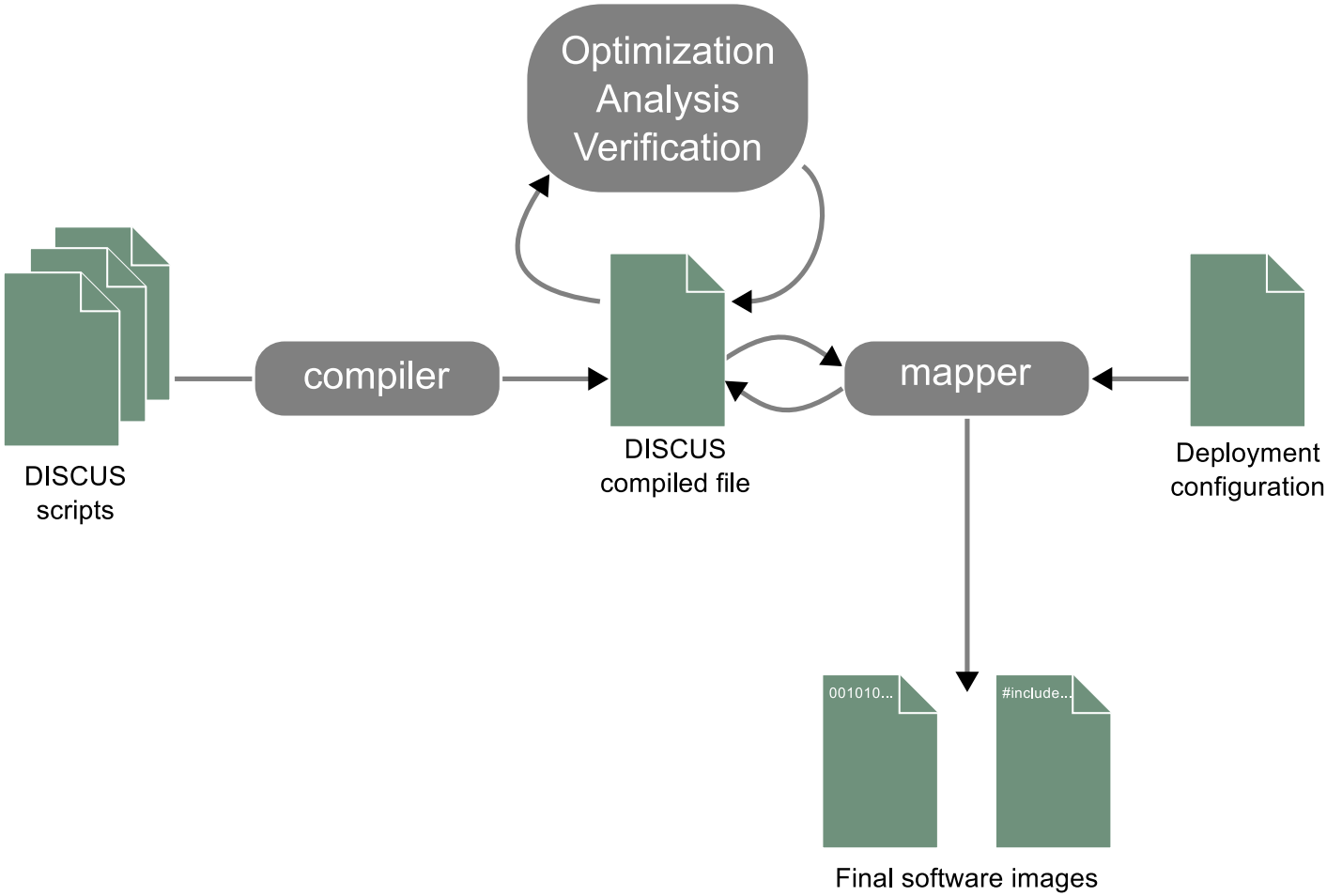
```
on dos_attempt(ipaddr src)
    for first entry in dos_attempt with
            entry.src == src
        update entry.count += 1
        update entry.last_attempt = now
        raise check_threshold(src)
    ifnone
        insert into dos_attempt {
            (...)
        };
```

Dealing with DOS attempts

```
on dos_attempt(ipaddr src)
    for first entry in dos_attempt with
            entry.src == src
        update entry.count += 1
        update entry.last_attempt = now
        raise check_threshold(src)
    ifnone
        insert into dos_attempt {
            (...)
        };

on check_threshold(ipaddr src)
    for first entry in dos_attempt with
            entry.src == src
            and entry.count >= 50
        alert("Some message");
```

DISCUS development process



Plan

What's done

- Compiler, optimizer
- Targets : embedded micro-controller, linux user space (libpcap), Snort scripts
- Automatic snort to discus translation : 98% of snort rules can be converted to DISCUS script
  - remaining 2% involves C code (snort modules)
- Simple data handling/sharing

Next steps

Enhanced data handling/sharing

- Efficient
- Secure
- Adapted to heterogeneous targets

Next steps

Enhanced data handling/sharing

- Efficient
- Secure
- Adapted to heterogeneous targets

More targets !

- FPGA !
  - High performance, quite low cost
  - Integrates IDS in switches
- Network cards firmwares

Next steps

Dynamic reconfiguration

- Redefine security rules
- Without stopping monitoring

More domains

- Industrial Control System Security
  - No more standard IP networks
  - Real time guaranties ?
- Securing IoT
  - Tiny IDS in every Thing

Questions ?

Michaël Hauspie – michael.hauspie@univ-lille1.fr

[1] D. N. M. Bryan et M. Anderson. *Cloud Computing, a Weapon of Mass Destruction ?* 2010. url : https://www.defcon.org/html/links/dc-archives/dc-18-archive.html.

[2] Ramana Rao Kompella. « On Scalable Attack Detection in the Network ». In : 2004.

[3] D. Riquet, G. Grimaud et M. Hauspie. « DISCUS : A massively distributed IDS architecture using a DSL-based configuration ». In : *Proceedings of the 2014 International Conference on Information Science, Electronics and Electrical Engineering (ISEEE'2014)*. Sapporo City, Hokkaido, Japan, 2014.

[4] D. Riquet, G. Grimaud et M. Hauspie. « Large-scale coordinated attacks : Impact on the cloud security ». In : *Proceedings of The Second International Workshop on Mobile Commerce, Cloud Computing, Network and Communication Security 2012 (MCNCS*