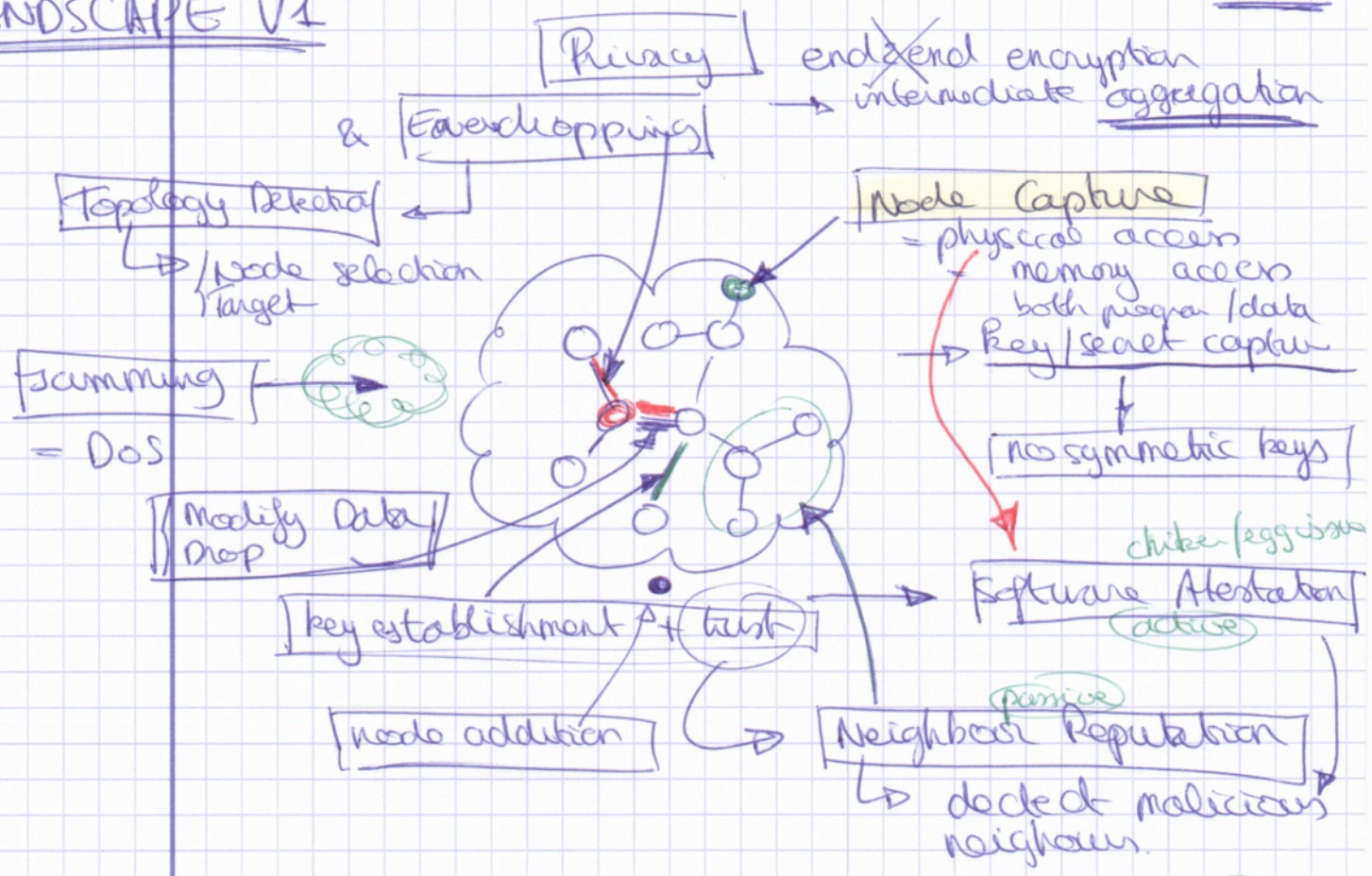
notes 1

LANDSCAPE V1



NOW active Intrusion → physical access to 1 node (HIDS) (local detection)

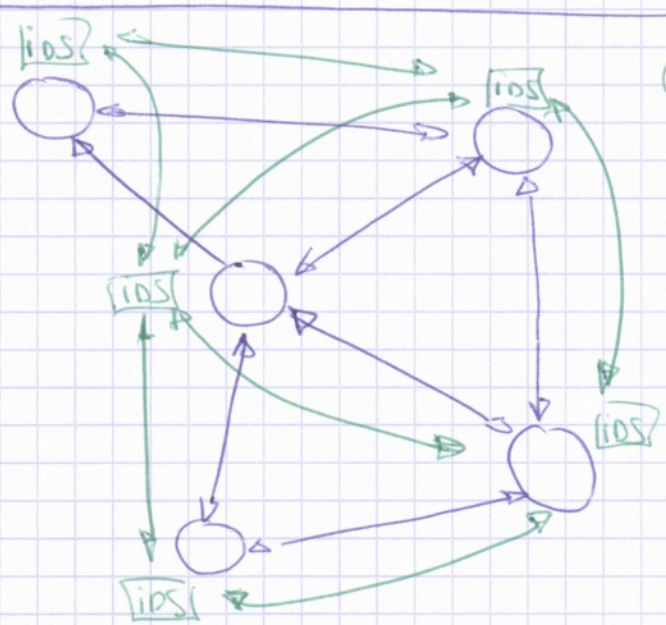
PAST re-active → compromise of 1 node in network. ← network detection (GIDS) / anomaly detection

FUTURE pro-active → over the wire access (NIDS)

+ node addition

! intelligent agent

! trust agents

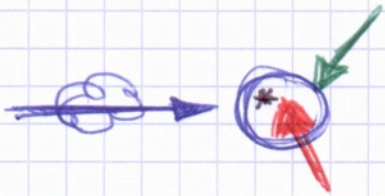


IDS overlays network.

agent – local
– remote exchange

CONCEPT V1

Notes &



FW

accept all kinds

- Network attacks
- Host attacks (~ physical)
- App attacks → very specific
- postmortem " → buffer overflow & CO
  ↳ it is done (SA) ↳ also brute force auth.

① event detection
② event correlation → anomaly detection
③ ↳ attack detection.
④ " reporting.

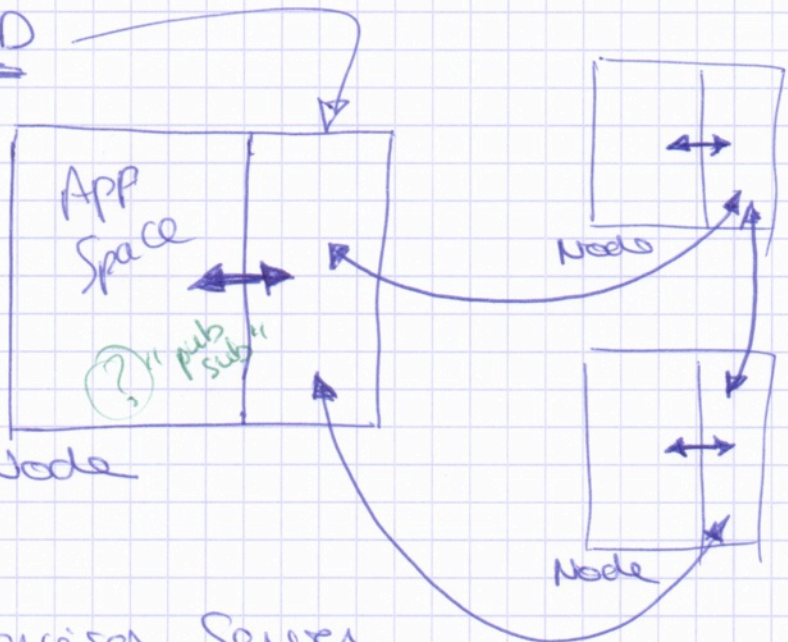
Node → | Network Group → server

LooCID

Loosely Coupled Intrusion Detection

:-)

≈ overlay network

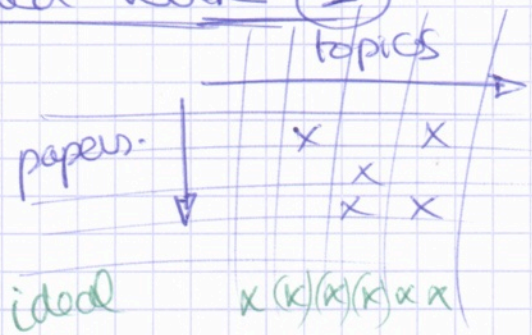


all communications are events.

FW (small)!! ← network proc
- comm
- auth
- pubsub
- correlate
- support all

+ Supervisor Server
↳ origin of "rules" | policy
↳ pushes through network
→ dynamic / group specific
(?) ↳ realtime response?
↳ to augment level
↳ divide network among special nodes

Matrix Related Work (1)



Thesis Structuur

Voorwoord / Dankwoord.
Samenvatting
Inleiding.
- Draadloze sensornetwerken
- Toepassingen ← VB !!
- Probleemstelling
- Doelstelling
- Verloop / Structuur tekst

Achtergrond.
- ~~...~~ (landscape, "notes" network, context, (cooci))
- Gerelateerd onderzoek
  - } major slices
  -

Probleemstelling ← Scenario's

2) Voorgestelde oplossing

Bestaande Technologieën
2
Architectuur
Implementatie
Discussie
Besluit

Demo

+ topology + supervisor

"simulate" 3 attacks ≤ 3 types

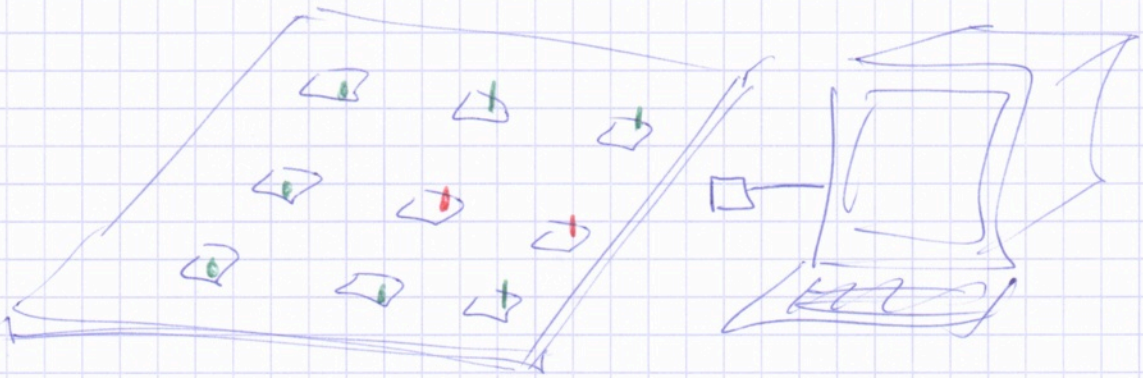
not implementable

network? application?

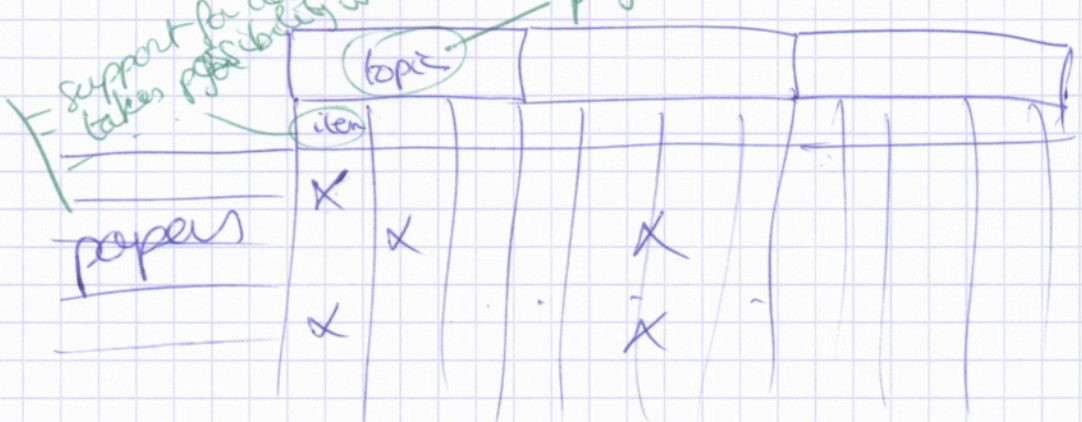
mind app? → with external interface

→ proximity? → marker detection

case "try not to be seen"



Malax related work (2) e.g. "physical"



Features

- end 2 end solution
- light weight
- non-intrusive (pub sub, event, ...)

report → Nodes ↓

node group ← GW - server

configure/install

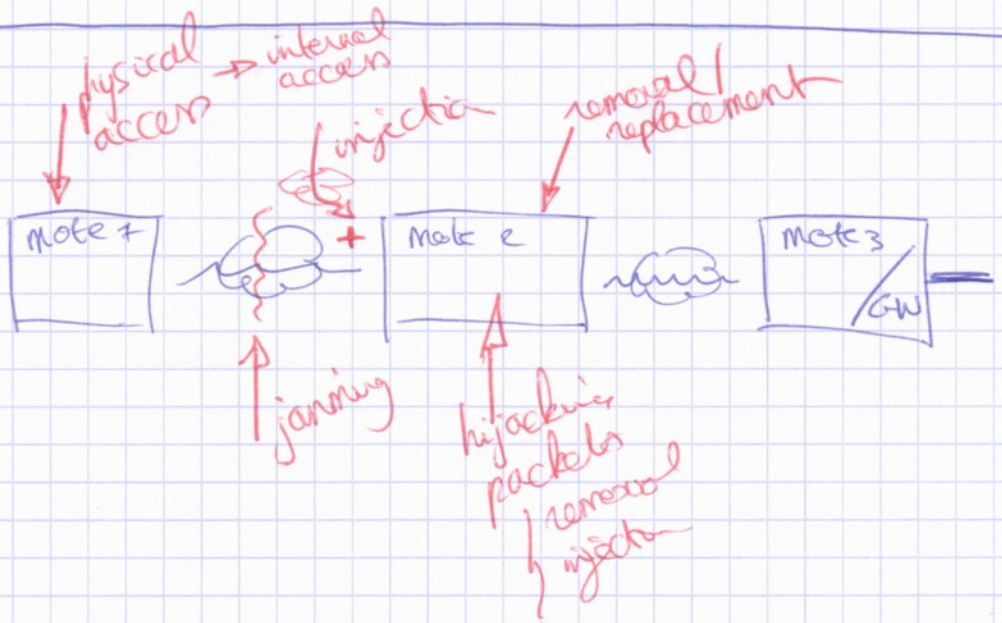
Reasons

- not in use — not in scope
  ↳ no offering

↳ detection while "sleeping" ? ...

↳ thesis NOT about detection = too large specific related work

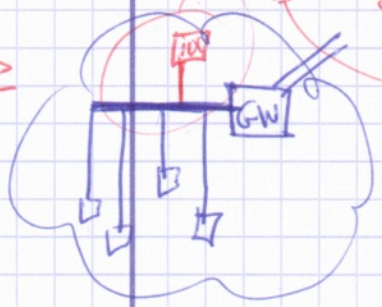
---



Notes 6

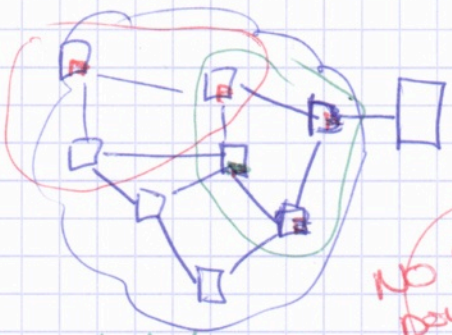
Differences "classic" IDS vs NSN-IDS

NIDS



"backbone" structure
see-all

vs

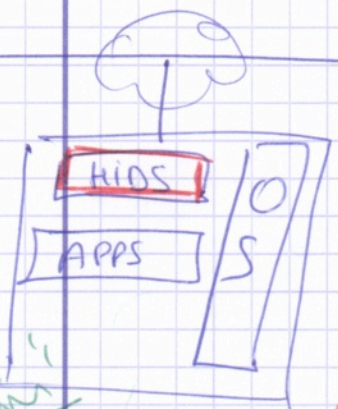


no single point of see-all

limited scope vision

=> no single instance possible

HIDS



"microsystem"

signatures

large list of events.

operating system "calls"

application specific

files focus. changes.

- END Node - ROUTER - coordinator
- Sensor Node - ClusterHead - sink

DSL hides system details → supporting unified centralized generic library

- request system info (local) → HOW? to abstract! → app specific info → memory addresses
- basic i/o "send"/"receive" → protocol "bytes"
  - ↳ between nodes
    - directive
    - informational → give request
    - remote "call"
  - ↳ listen promiscuous → "costly"
    - ↳ "a packet"? or (abstracted data format)
- R/W store (structured byte array in mem) + offset, constants → for computations → DSL access / optimized mem storage/cycles
  - → reuse of code
  - → "lighter" api (generic)
  - ↳ optimized storage with high level API
    - + fast increment/decrement
    - + event based "rule engine"
      - e.g. alert when counter x reaches y

DSL at dev time ↳ compiles optimal runtime code → OS - ignorant / independent → optimization

↓ POC

test of access hardcoded byte array is faster/smaller than "classic" code with nice interface.

↳ need literature ← example?! (DSL more optimal code than rewriting ??)

@ gateway / controller / supervisor

DSL → 1/ translate (compile) to "optimized size / cpu" C-code } MEO (possible improved?) "optimality" vs. classic API Meg.

2/ compile to (hex)

3/ OTAP onto nodes, according to policy

→ node-capabilities { based / dependent

---

optimize multirules (in classic code this would be a no-go ⇒ improved language..)

↳ functional

DSL → (language rewriting)⁺ → C... → direct to ASM? ≈ out-do compiler :-) → avoid costly type checks?

"classic signatures" ↳ become DSL { pattern / anomaly / directive

→ event based

1/ counterX ++

2) when counterX > 10 → "actions" / commands

DEV: DSL → C → OTAP → Node MOCCAS / FW → loop controller / OS (if any)

supervisor or gateway

minimal cmd structure

no loops!

1/ condition

2/ assignment (operator) → command.

3/ event trigger definition

4/5/ ~~request~~ set/get "info" → local system / remote / computed / messaging.. ↳ implicit

DSL Cisco = ISDN SL (intrusion detection specification language :-))

node { shouldn't / doesn't } know about others → "might not be there" (abstracted)

↳ check if possible to implement algo's that DO access specific other nodes' info.

notes 9

- DSL
- code generation } show possibility justify
- FW + API → very low-level technical.

LooCiD FW

Loosely Coupled Intrusion Detection FW

- Contiki (+ LooCi)
  - "or not"

(?) event driven ? DSL

(tooling) just may not be optimal path

thesis "scope"

subset possibilities → no DSL

just enough to prove overall possibilities

based on "detection"

literature →

1) detection WSN
2) detection classic.
3) DSL → code generation

Demo // show classic implementation
X 2-3 detections
versus (generated) low-level API + FW
show workings
+ code/mem/cpu reduction

attach selection → "visibility" ① "Sybil" ?
② "sink hole" ?
la mama ③ "node capture" ?
anomalie.

:->

notes 10/14

! limit max size = compiler policy

mem = compromise to CPU

hidden through DSL.

choose.

= battery

use more = speed = more lifetime

(reuse space)

reserved mem block

e.g. cyclic buffer size

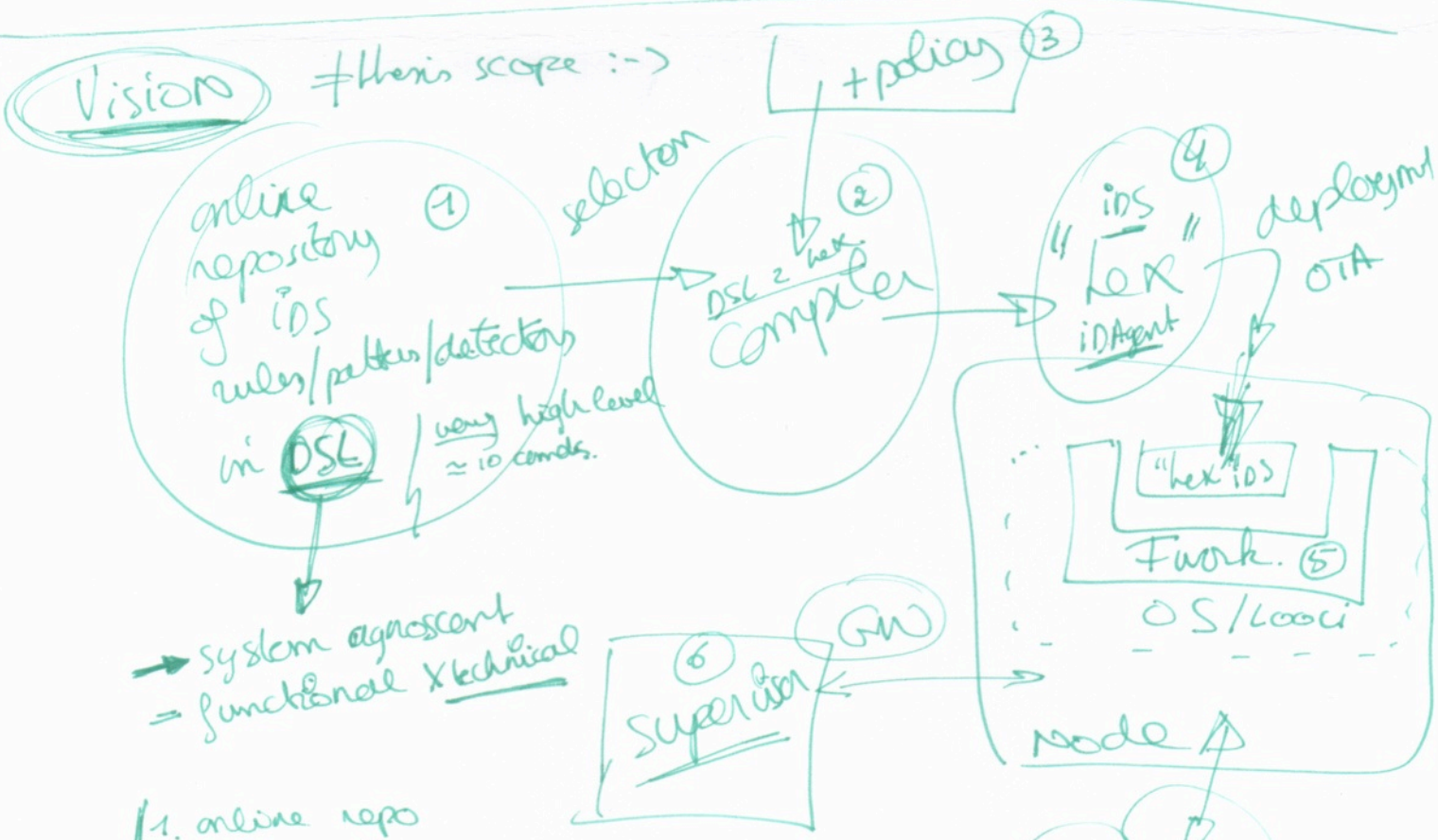
"transparent" ≈ conversion (functions)

↳ "encrypt" ?+

↳ subblock exchange.

possibilities show/pray X implement all ?

Vision ≠ this is scope :->



→ system agnoscent

= functional X technical

1. online repo
2. DSL compiler
3. policy
4. low level API implementation.
5. supported API framework
6. supervision tools

+ scope of Hardware → ATMEGA1284P + Contiki + Loosi