CyberExcellence-2022

UCLouvain

ÉCOLE POLYTECHNIQUE DE LOUVAIN

SEMA

Symbolic Execution toolchain for Malware Analysis - **Packing**

By ***Christophe Crochet*** & ***Charles-Henry Bertrand Van Ouytsel*** & ***Khanh Huu The Dam*** & ***Serena Lucca***
Under the supervision of ***Axel Legay***

UCLouvain

Malware analysis to defeat them all

Symbolic Execution you said ?

SEMA

Packing is kinda a problem

What's next ?

ÉCOLE POLYTECHNIQUE DE LOUVAIN

UCLouvain

Malware analysis to defeat them all

Symbolic Execution you said ?

SEMA

Packing is kinda a problem

What's next ?

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Malware analysis to defeat them all

Malware

= "*Malware is a piece of code which changes the behavior of either the operating system kernel or some security sensitive applications, without a user consent and in such a way that it is then impossible to detect those changes using a documented features of the operating system or the application (e.g. API).*" - Introducing Stealth Malware Taxonomy

Malware Analysis

= Process to understand behavior of suspicious program

Malware analysis techniques

Static analysis

= Malware analysis based on syntaxic properties defining a signature

Example of tool: Yara

Dynamic analysis

= Malware analysis based on program execution

Example of tool: volatility

Malware analysis techniques problems

Static analysis

- Easily tricked with variants
- With encryption/packing
- Example: detecting string "I'm evil"

```
ULONGLONG uptime = GetTickCount();
Sleep(500000);
ULONGLONG uptimeBis = GetTickCount();
if ((uptimeBis - uptime)<500000 || IsDebuggerPresent()){
   MessageBox(NULL,"Hello world!", "", MB_OK);
} else{
    char message[20] = "";
    HINSTANCE hlib = LoadLibrary("msvcrt.dll");
    MYPROC func = (MYPROC) GetProcAddress(hlib,
                                          "strcat");
    (func) (message, "I'm ");
    (func) (message, "evil!!");
    MessageBox(NULL, message, "", MB_OK);
 }
```

Malware analysis techniques problems

Dynamic analysis

- Anti-debugger
- Time constraints
- ...

- Example: detecting string "I'm evil"

```
ULONGLONG uptime = GetTickCount();
Sleep(500000);
ULONGLONG uptimeBis = GetTickCount();
if ((uptimeBis - uptime)<500000 || IsDebuggerPresent()) {
   MessageBox(NULL,"Hello world!","", MB_OK);
} else {
    char* fl[2] = {"cat","str"};
    char buf[10],message[20];
    strcpy(buf, fl[1]);strcat(buf, fl[0]);
    HINSTANCE hlib = LoadLibrary("msvcrt.dll");
    MYPROC func = (MYPROC) GetProcAddress(hlib, buf);
    (func) (message, "I'm "); (func) (message, "evil!!");
     MessageBox(NULL, message, "", MB_OK);
}
```

Malware analysis techniques problems

Dynamic analysis

- Anti-debugger
-
-
-

Solution

Symbolic Execution

UCLouvain

Malware analysis to defeat them all

Symbolic Execution you said ?

SEMA

Packing is kinda a problem

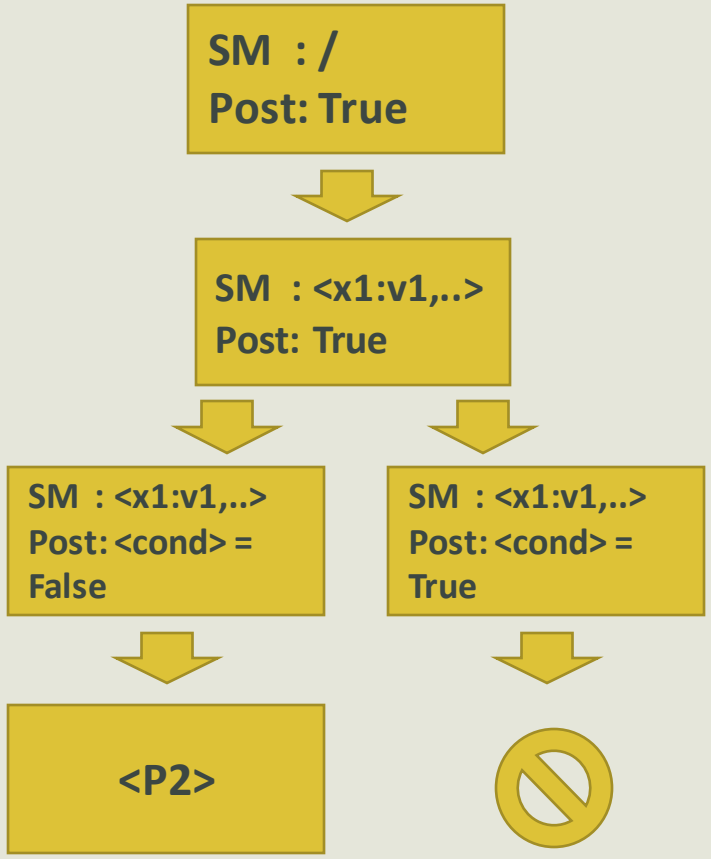
What's next ?

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Symbolic Execution you said ?

- Program execution of all possible paths (in theory)
  - Symbolic execution engine
- Symbolic memory store (*SM*)
  - For symbolic value &
  - Symbolic expression
- SMT solver use for satisfiability during path exection (post/pre)

```
<INIT>
if (<CONDITION>) {
    <exit>
 } else{
    <P2>
 }
```



Symbolic Execution you said ?

- Program execution of all possible path
  - Symbolic execution en
- Symbol
- S
  (p

Solve previous problems

BUT

Path explosion problem

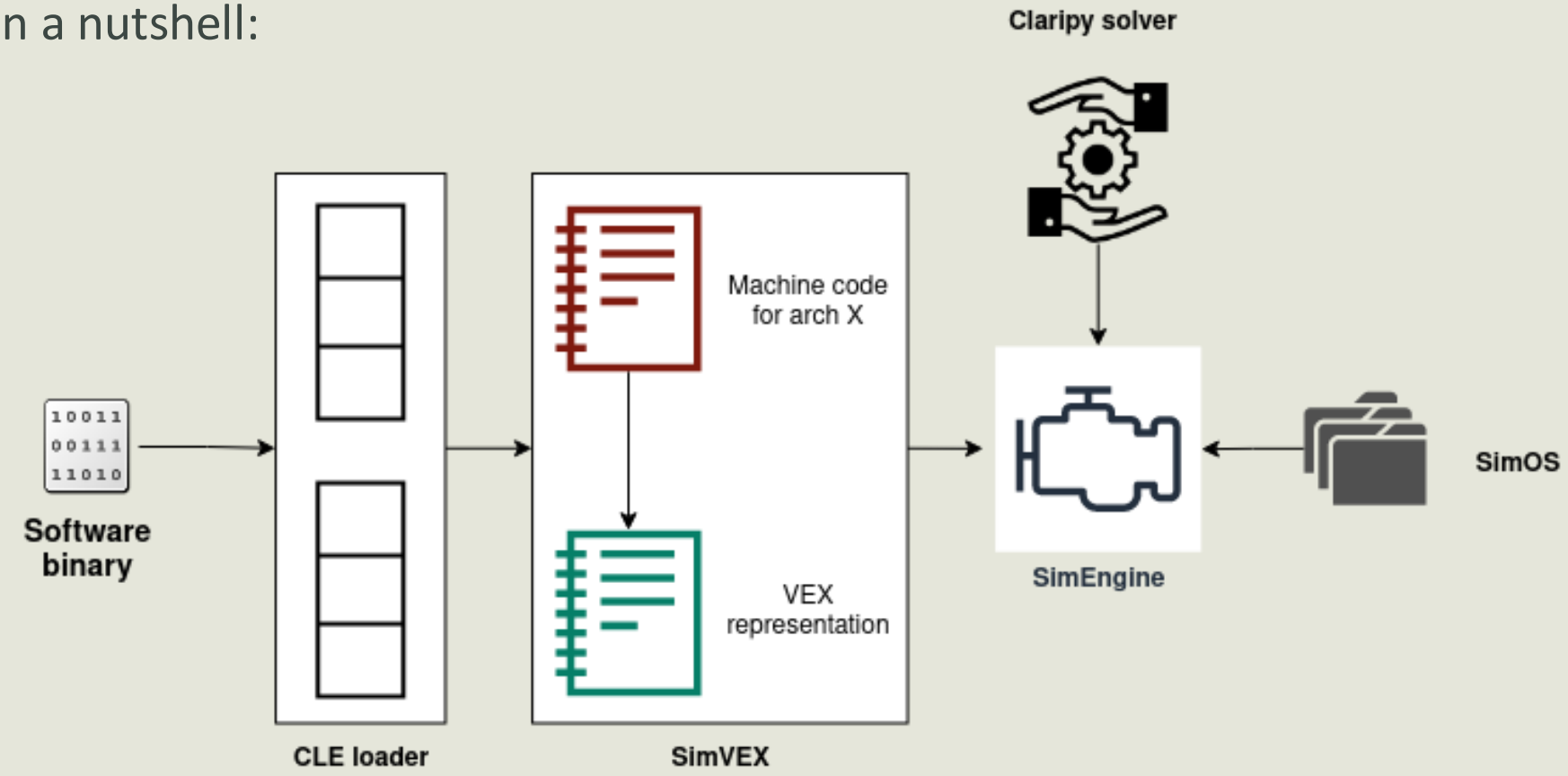
Angr(rr)

- "Open-source binary analysis platform for Python"
- Designs goals:
  1. Cross-architecture support
  2. Cross-platform support
  3. Multiple analysis paradigms support
  4. Usability

(State of) The Art of War: Offensive Techniques in Binary Analysis
Yan Shoshitaishvili, Ruoyu Wang, Christopher Salls, Nick Stephens, Mario Polino, Audrey Dutcher, John Grosen, Siji Feng, Christophe Hauser, Christopher Kruegel, Giovanni Vigna

Angr(rr)

Angr in a nutshell:



UCLouvain

Malware analysis to defeat them all

Symbolic Execution you said ?

SEMA

Packing is kinda a problem

What's next ?

ÉCOLE POLYTECHNIQUE DE LOUVAIN

SEMA

- Open-source project too !

- Build on top of Angr

- Goals:
  1. Malware detection
  2. Malware classification
  3. Collaborative works
  4. System calls graph (SCDGs) based analysis

SEMA

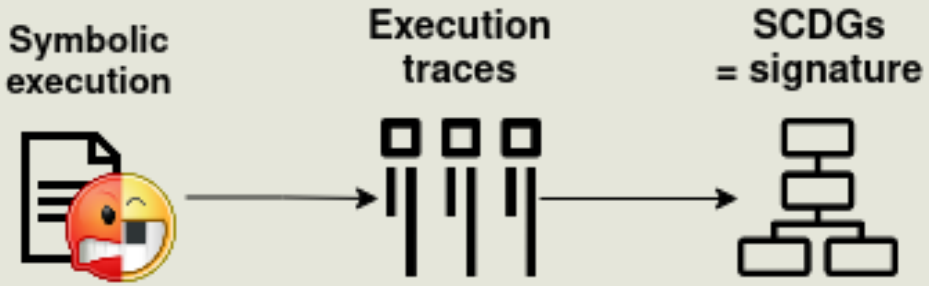
SEMA in a nutshell:

1. SEMA-SCDGs
2. SEMA-Classifier
3. SEMA-FL

SEMA

SEMA in a nutshell:

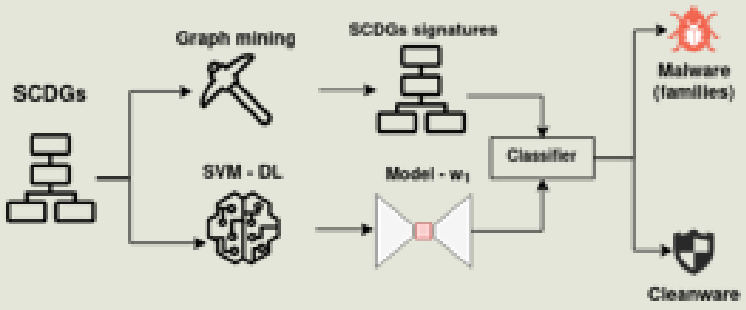
- SEMA-SCDGs
  - ELF & PE programs
  - Custom explorations techniques (CDFS & CBFS)
  - Track of executions paths with SCDGs



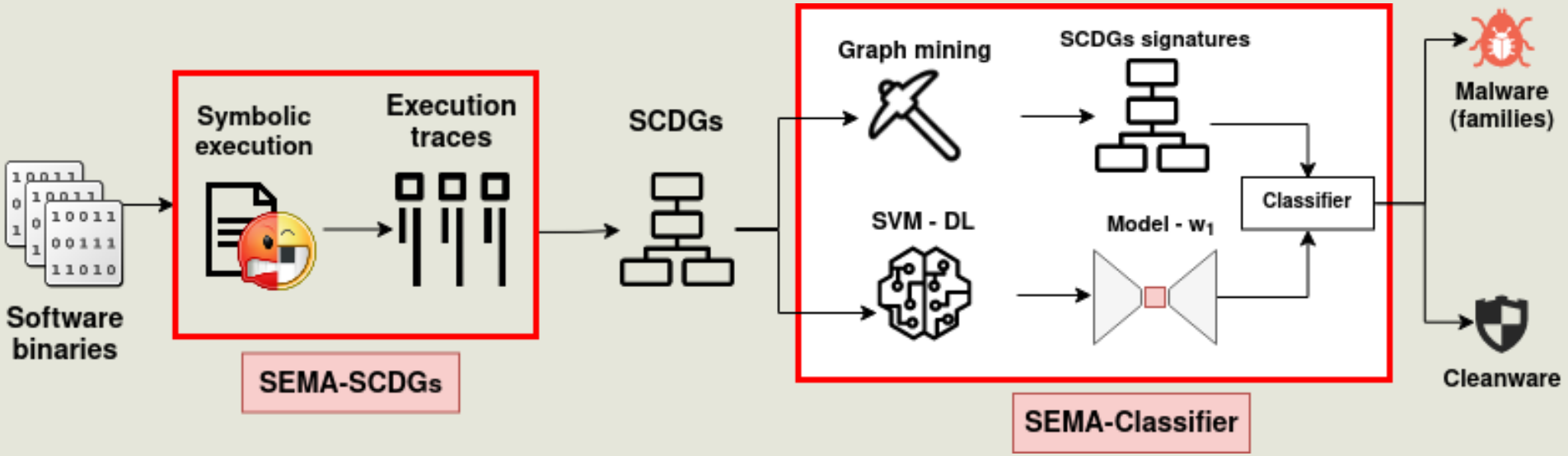
SEMA

SEMA in a nutshell:

- SEMA-Classifier
  - Use SCDGs produced as signature
  - Graph mining model (gSPAN)
  - SVM with graph kernel model
  - Deep learning model



Standalone

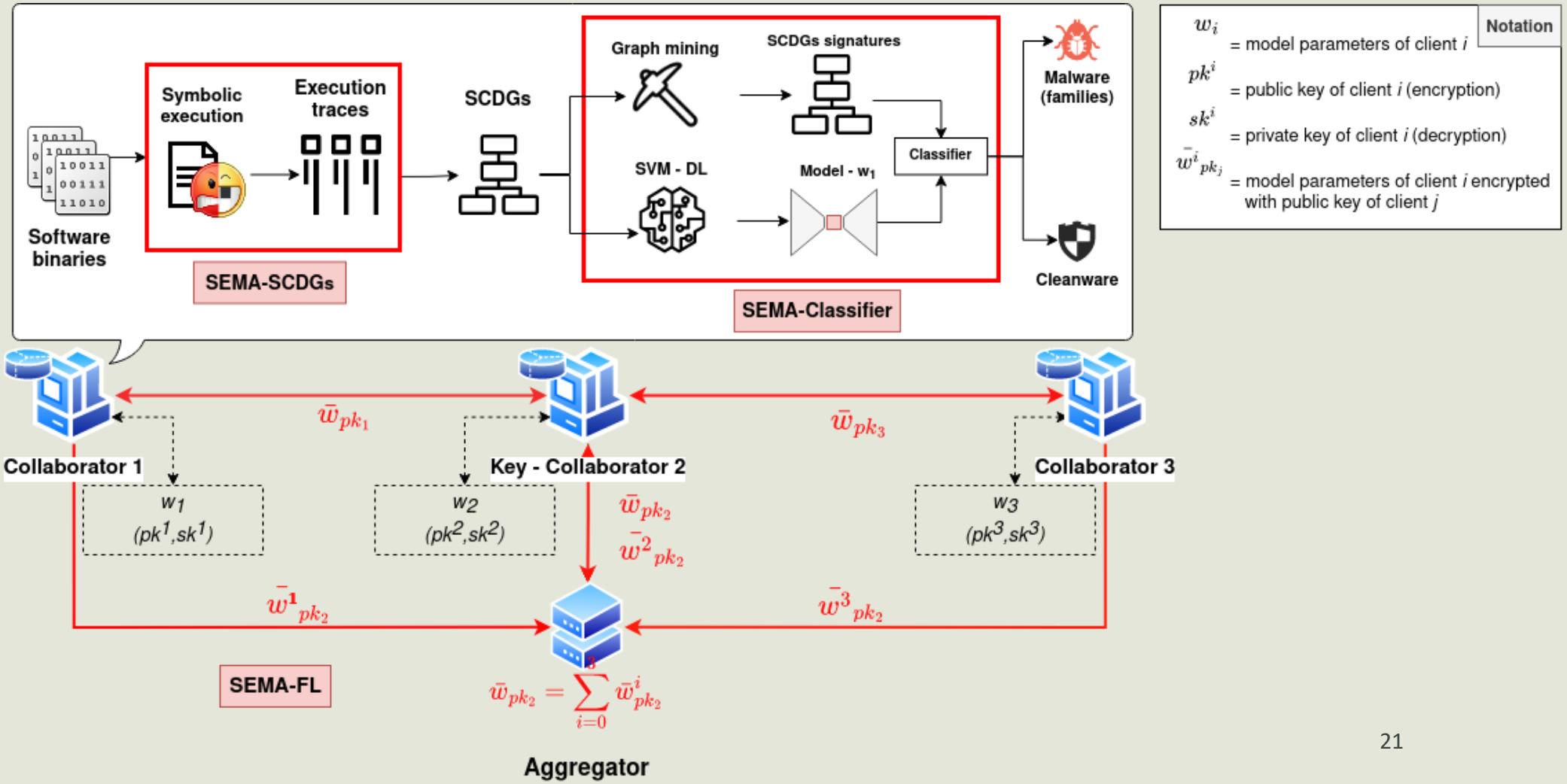

Software binaries
Symbolic execution
Execution traces
SEMA-SCDGs
SCDGs
Graph mining
SCDGs signatures
SVM - DL
Model - w_1
Classifier
SEMA-Classifier
Malware (families)
Cleanware

SEMA

SEMA in a nutshell:

- SEMA-FL
  - Trust server model
  - N clients with their own database
  - Only deep learning model
  - Homomorphic encryption for shared parameters

Adding Federating Learning



UCLouvain

Malware analysis to defeat them all

Symbolic Execution you said ?

SEMA

Packing is kinda a problem

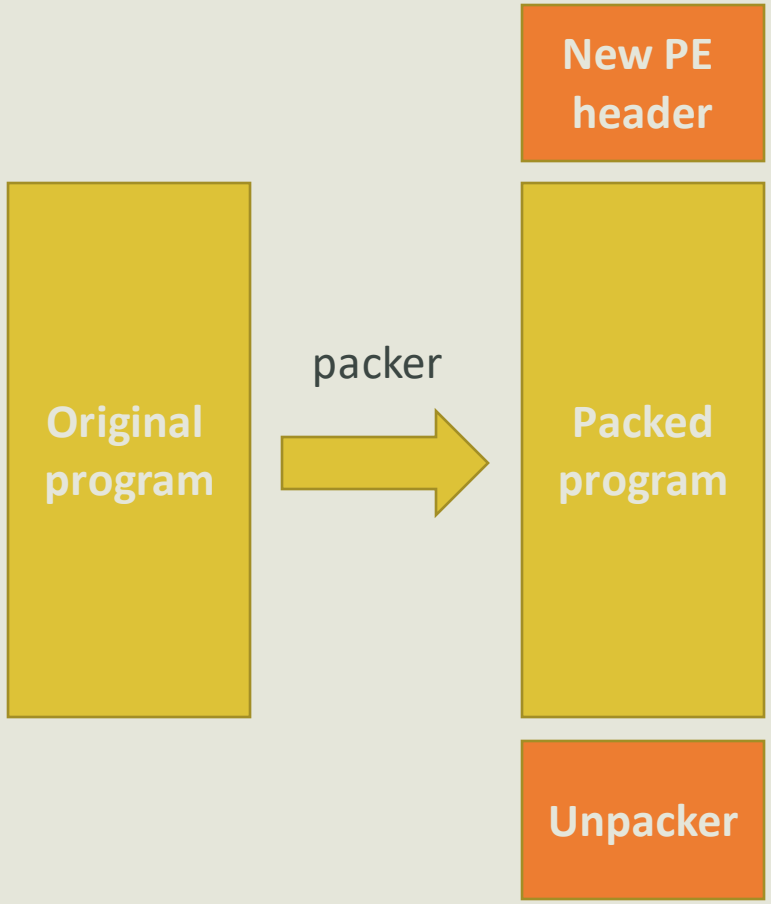
What's next ?

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Packing is kinda a problem

What is packing ?

- Obfuscation technique use to hide original program
  - Formatting, compression, etc
- Stub routine to unpack the original code
- E.g: UPX, PE-packer, etc.



Packing is kinda a problem

What is packing ?

- Obf

- S

- E.g

Solution

Concolic Execution

Concolic Execution

Idea = Concolic Execution with Symbion

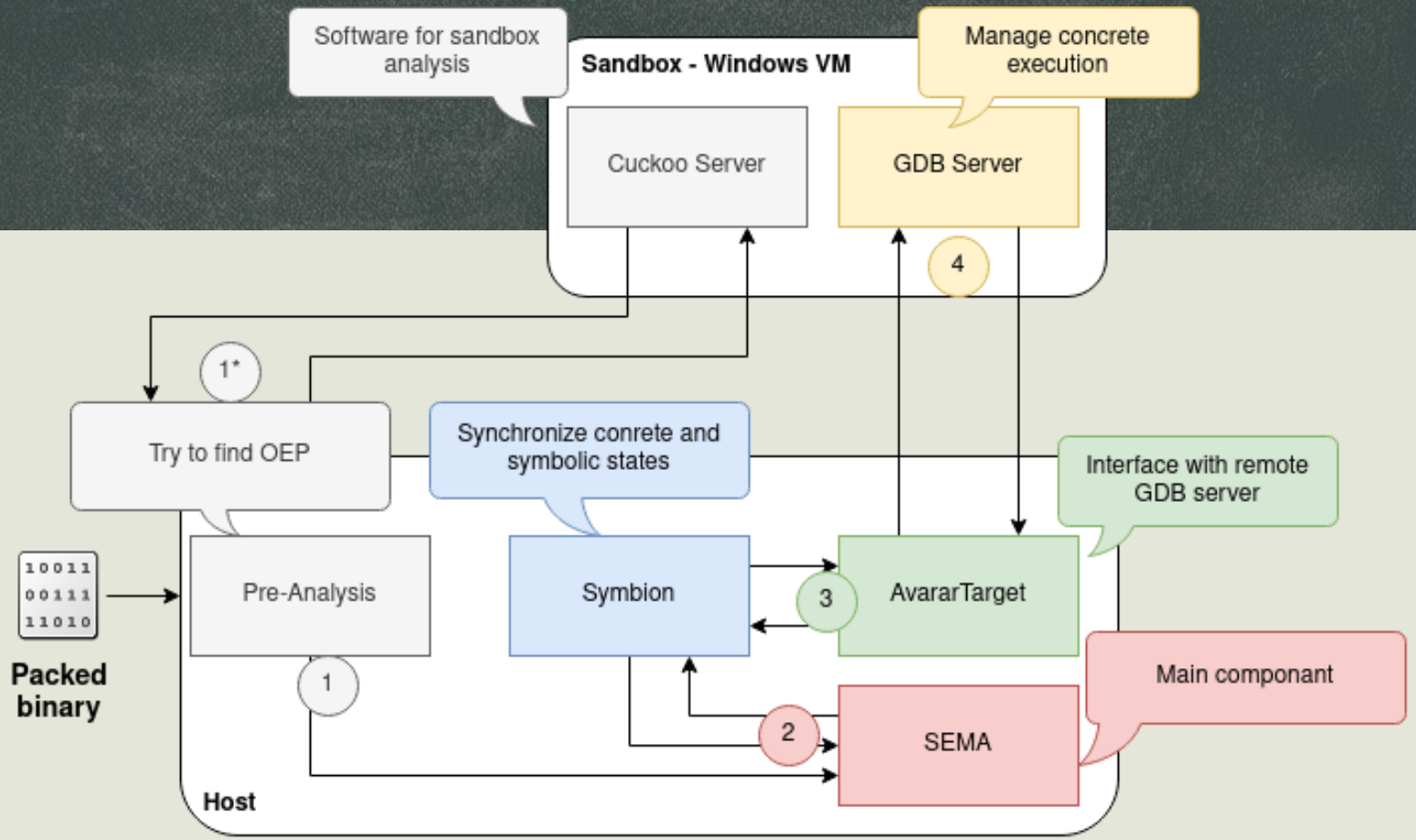
- Execute concretly the unpacking routine
- Execute symbolically the original malware

Challenges:

- Find original entry point of the malware
- Synchronize the state after concrete execution
- Dealing with modified headers

More concretly...

- Memory dumping for multi-layer packer
- Header reconstruction



1 **Send:** OEP if found + other informations to SEMA

2 **Send:** Address to stop concrete execution to Symbion
Receive: Synchronized state of desired address from Symbion

3 **Send:** Concrete output from steps to Symbion
Receive: Symbolic steps from Symbion

4 **Send:** Concrete output from command to Target
Receive: GDB command from Target

UCLouvain

Malware analysis to defeat them all

Symbolic Execution you said ?

SEMA

Packing is kinda a problem

What's next ?

ÉCOLE POLYTECHNIQUE DE LOUVAIN

What's next ?

- Extend federated learning to all models
- Support new types of programs (.NET, Java, Macros Excel, …)
- Extend exploration techniques
- Manage packed programs
- Manage obfuscation techniques

Concolic execution

- Many more

What's next ?

- Extend federated learning to all mod
- Support new typ
-
-
-
- M

All in progression !!

What's next ?

- Extend federated learning to all mod
- Support new typ
-
-
-
- M

Checkout our git

https://github.com/csvl/SEMA-ToolChain

CyberExcellence-2022

UCLouvain

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Toward Formal Specification of QUIC attackers with IVy

By ***Christophe Crochet*** & ***Tom Rousseaux***
Under the supervision of ***Axel Legay***

UCLouvain

QUIC is the future

Methodology for the formal verification of QUIC

Previous work

Attacker model

ÉCOLE POLYTECHNIQUE DE LOUVAIN

UCLouvain

QUIC is the future

Methodology for the formal verification of QUIC

Previous work

Attacker model

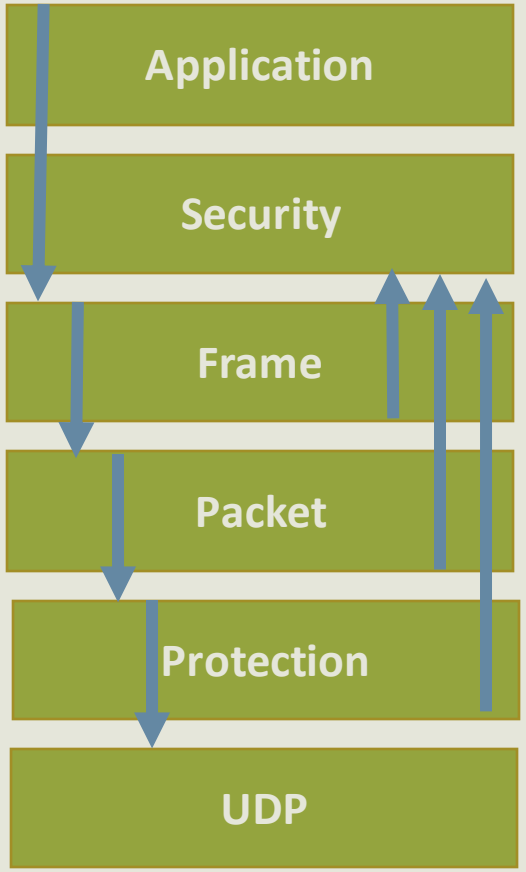
ÉCOLE POLYTECHNIQUE DE LOUVAIN

QUIC is the future

- QUIC: a new **secure** transport protocol
  - Intended to replace TCP
  - RFC9000 = textual document

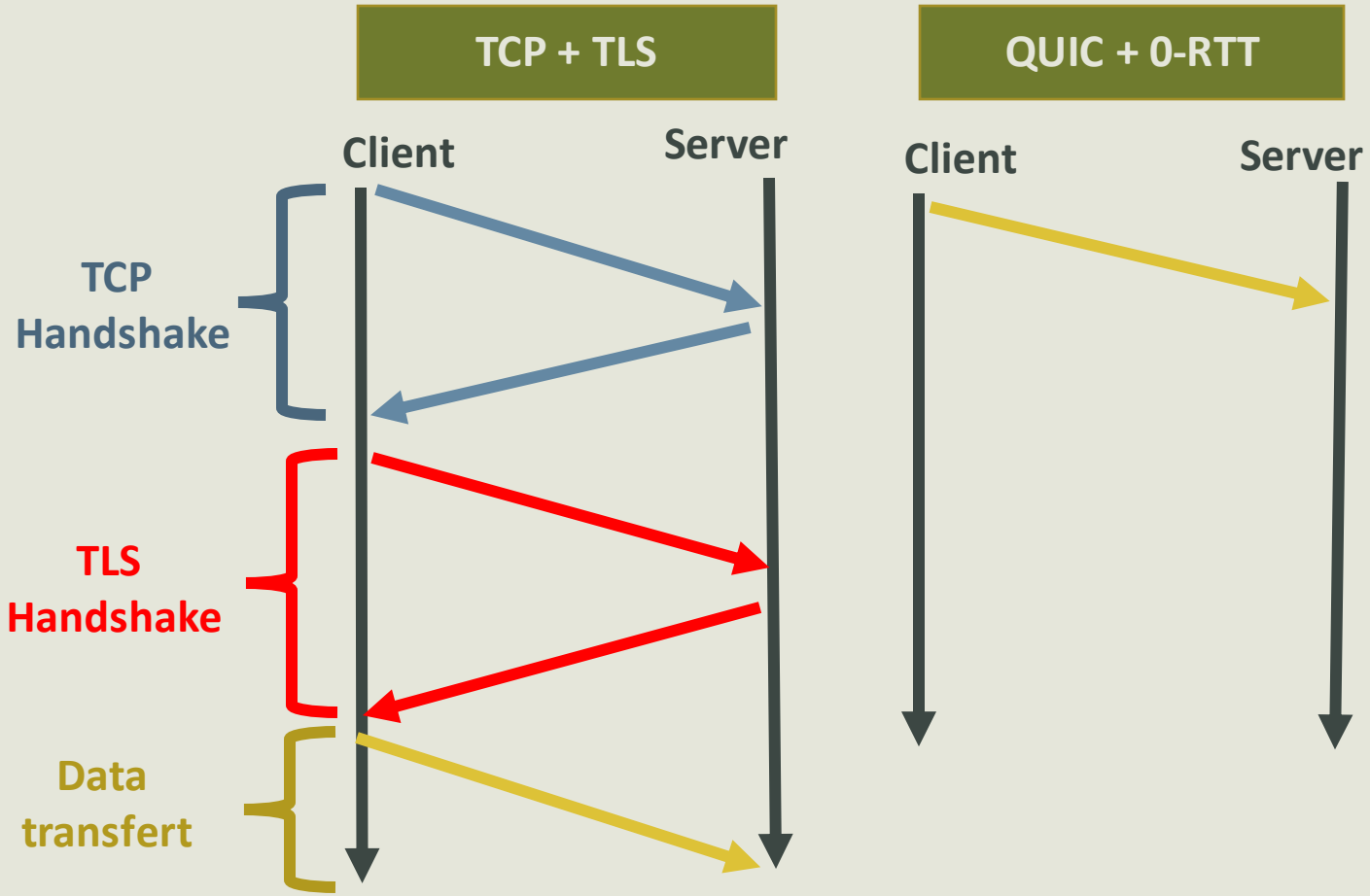
- Importance to test compliance of QUIC to its specification

- Formal verification versus interoperability tests



QUIC, a protocol with innovative features

- QUIC connection
- QUIC multiplexing
- QUIC migration
- Extensibility



UCLouvain

QUIC, a protocol with innovative features

Methodology for the formal verification of QUIC

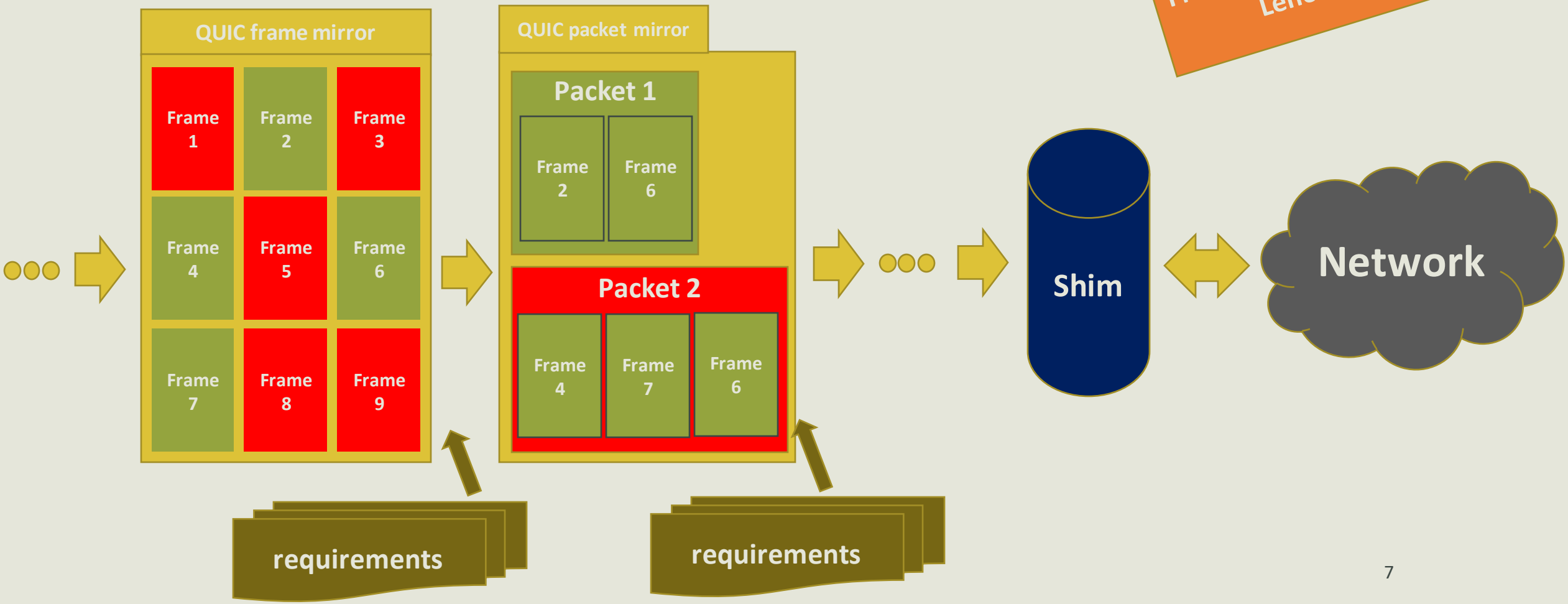
Previous work

Attacker model

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Randomized and Network-centric Compositional testing

From Kenneth McMillan and Lenore D. Zuck



UCLouvain

QUIC, a protocol with innovative features

Methodology for the formal verification of QUIC

Previous work

Attacker model

ÉCOLE POLYTECHNIQUE DE LOUVAIN

What we done

- Update the model to RFC9000 (from draft 18)
- Errors found in every implementation
  - Tested on 8 implementations
- Problems in the draft detected
  - Ambuiguities
- One implementation improved

Main problems founds

1. Violation of the specification
2. Feature not implemented
3. Internal errors and crashes
4. Problem in the draft

35 main errors developed

UCLouvain

QUIC, a protocol with innovative features

Methodology for the formal verification of QUIC

Previous work

Attacker model

ÉCOLE POLYTECHNIQUE DE LOUVAIN

Attack model

- Instead of formally specify QUIC protocol from RFC9000

- We formally specify "Man in the Middle" attacker of QUIC

Difficulties:

- No clear specification
- Localhost
- Usually attacks are very specific

Attack model

- Instead

- U very specific

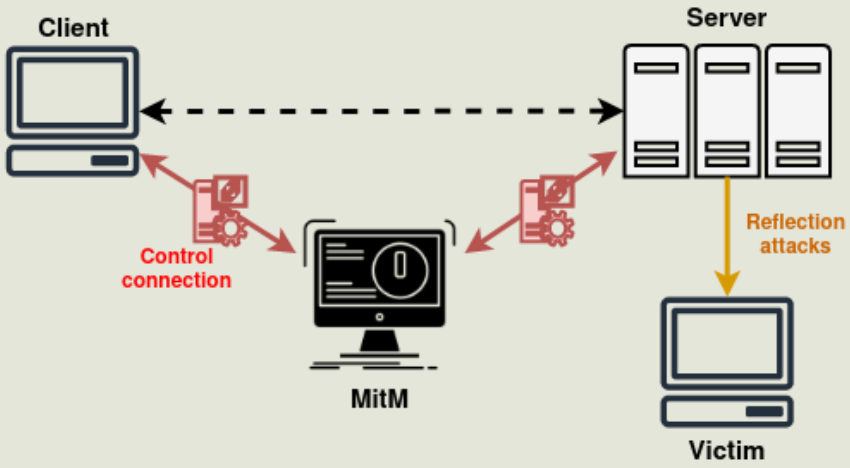
Template model

+

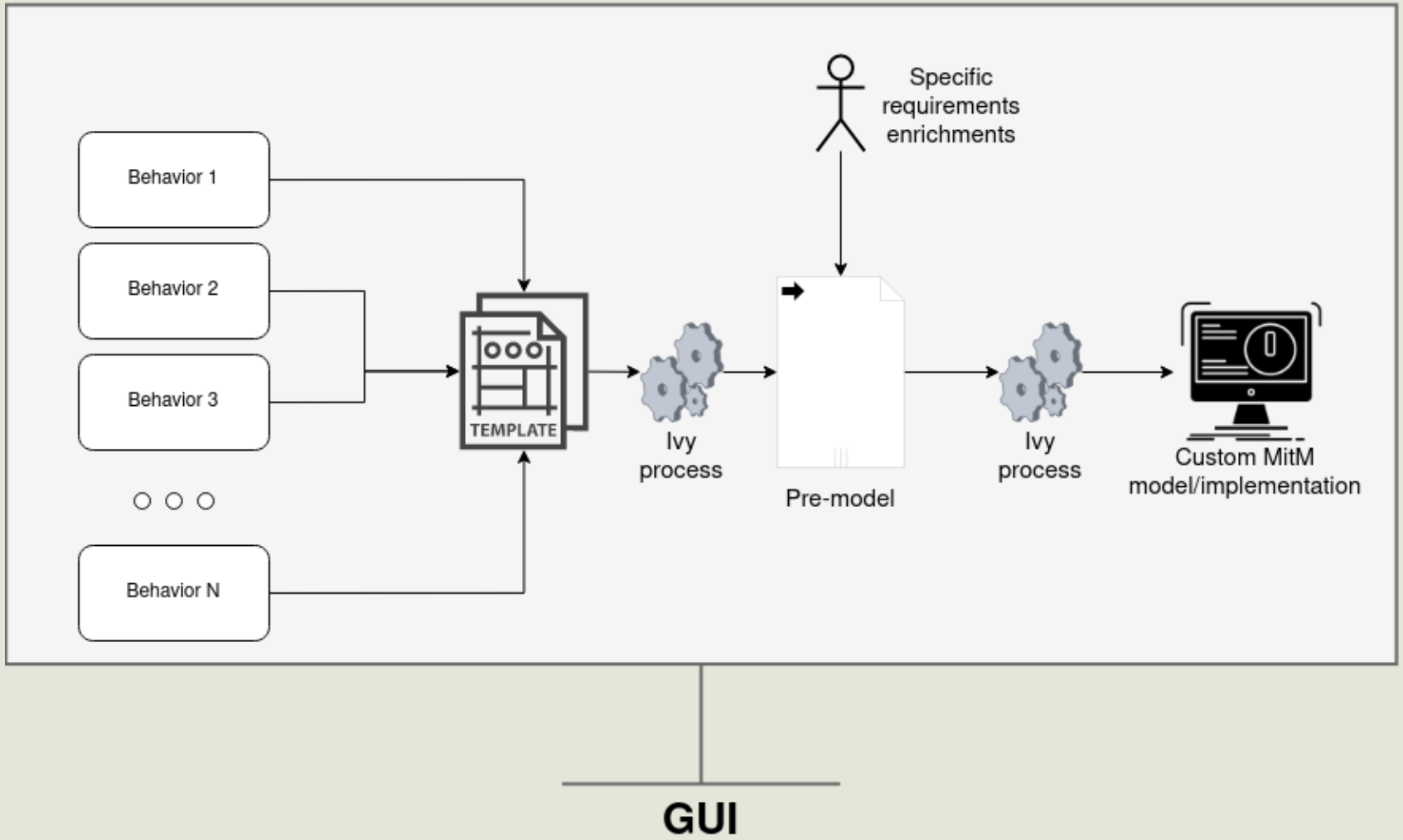
Simulator

Man in the Middle

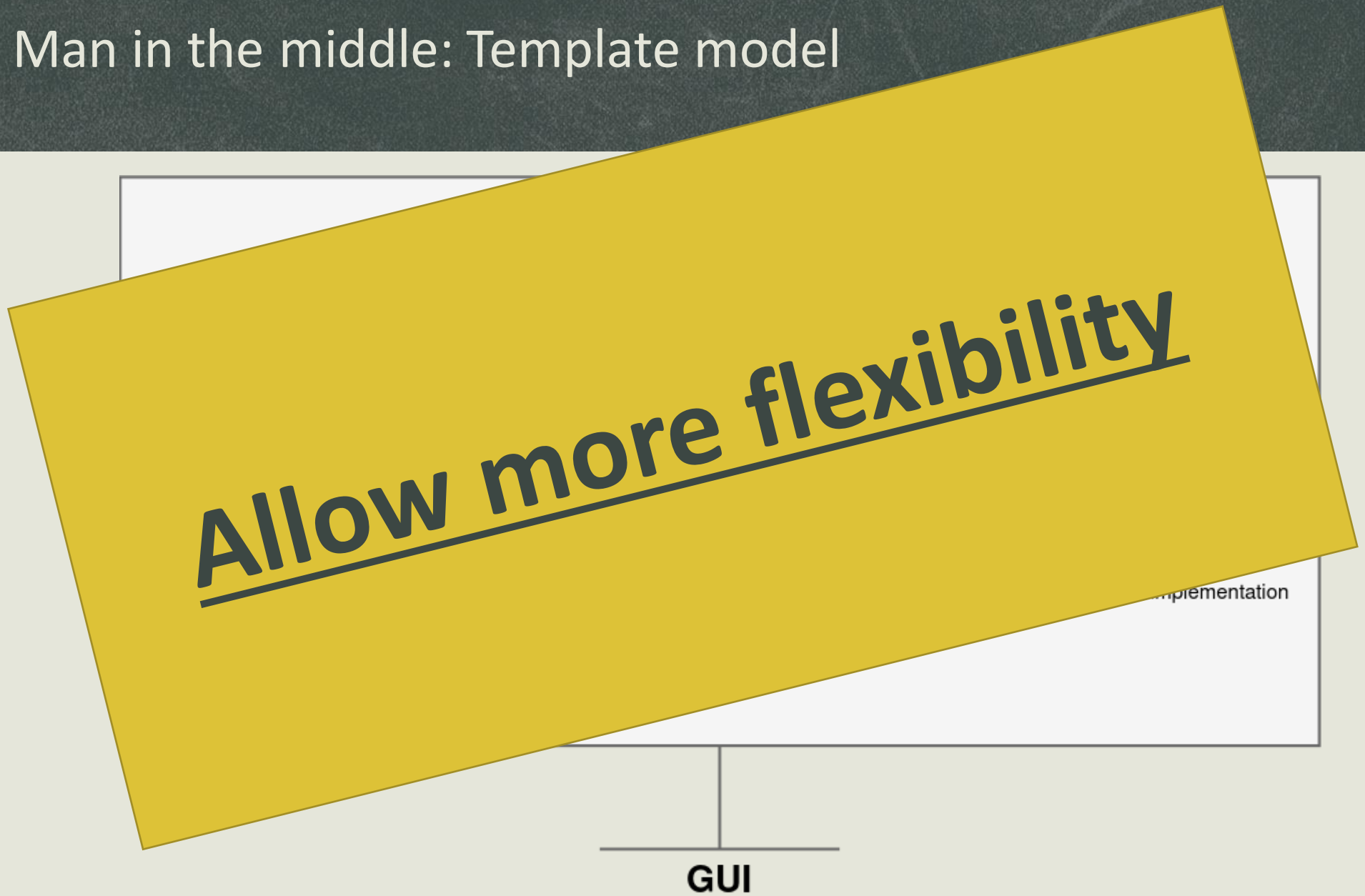
- MitM =
  - attacker placed between communication(s)
  - Able to listen/alter the communication(s)
  - Endpoints are not conscious of the attacker



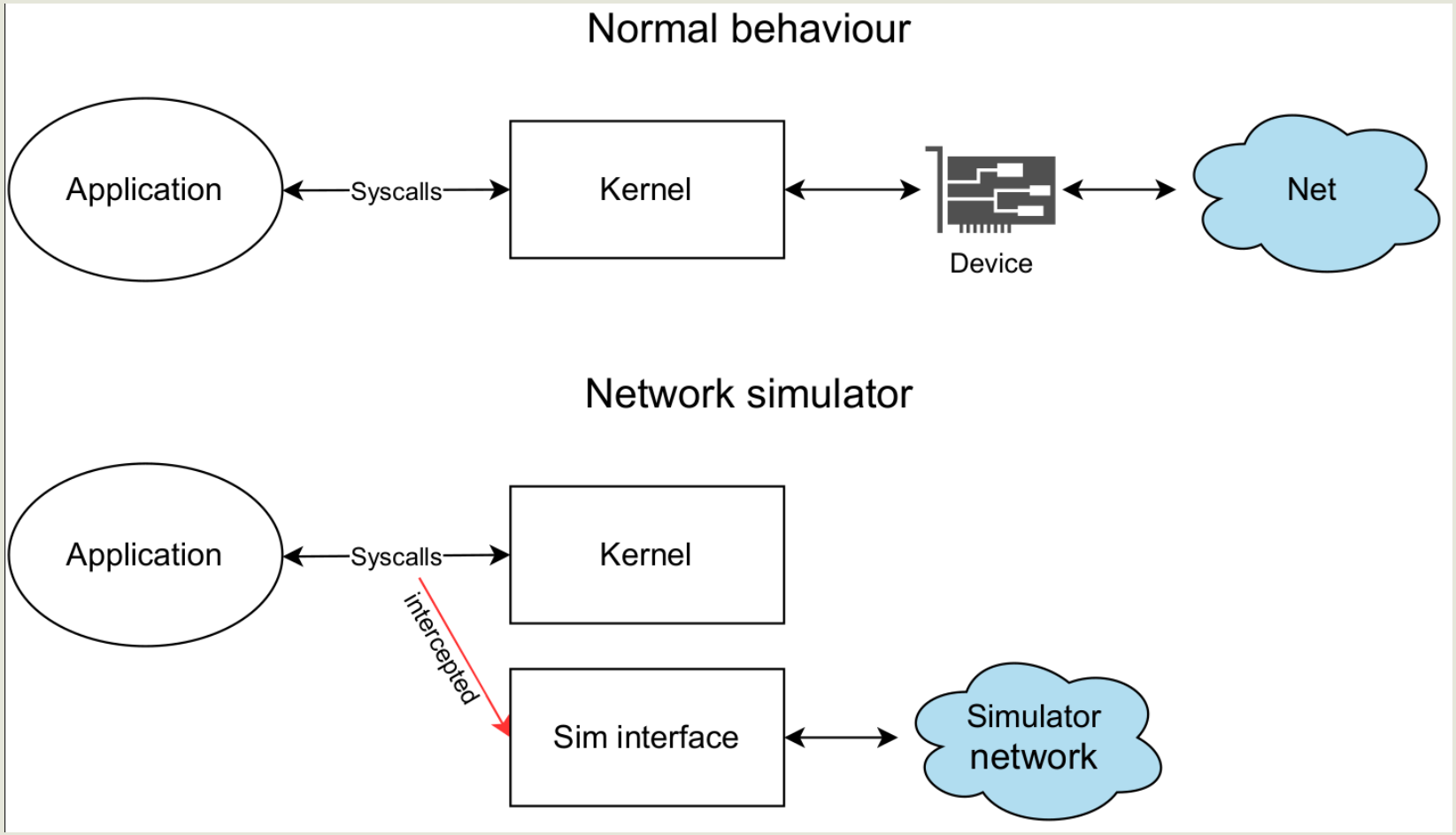
Man in the middle: Template model



Man in the middle: Template model



Simulator


Normal behaviour
Application
Syscalls
Kernel
Device
Net
Network simulator
Application
Syscalls
Kernel
intercepted
Sim interface
Simulator network

Simulator

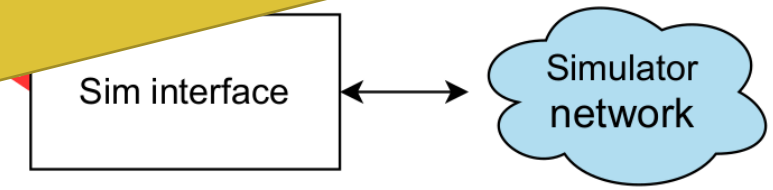
Reproductible experiments

+

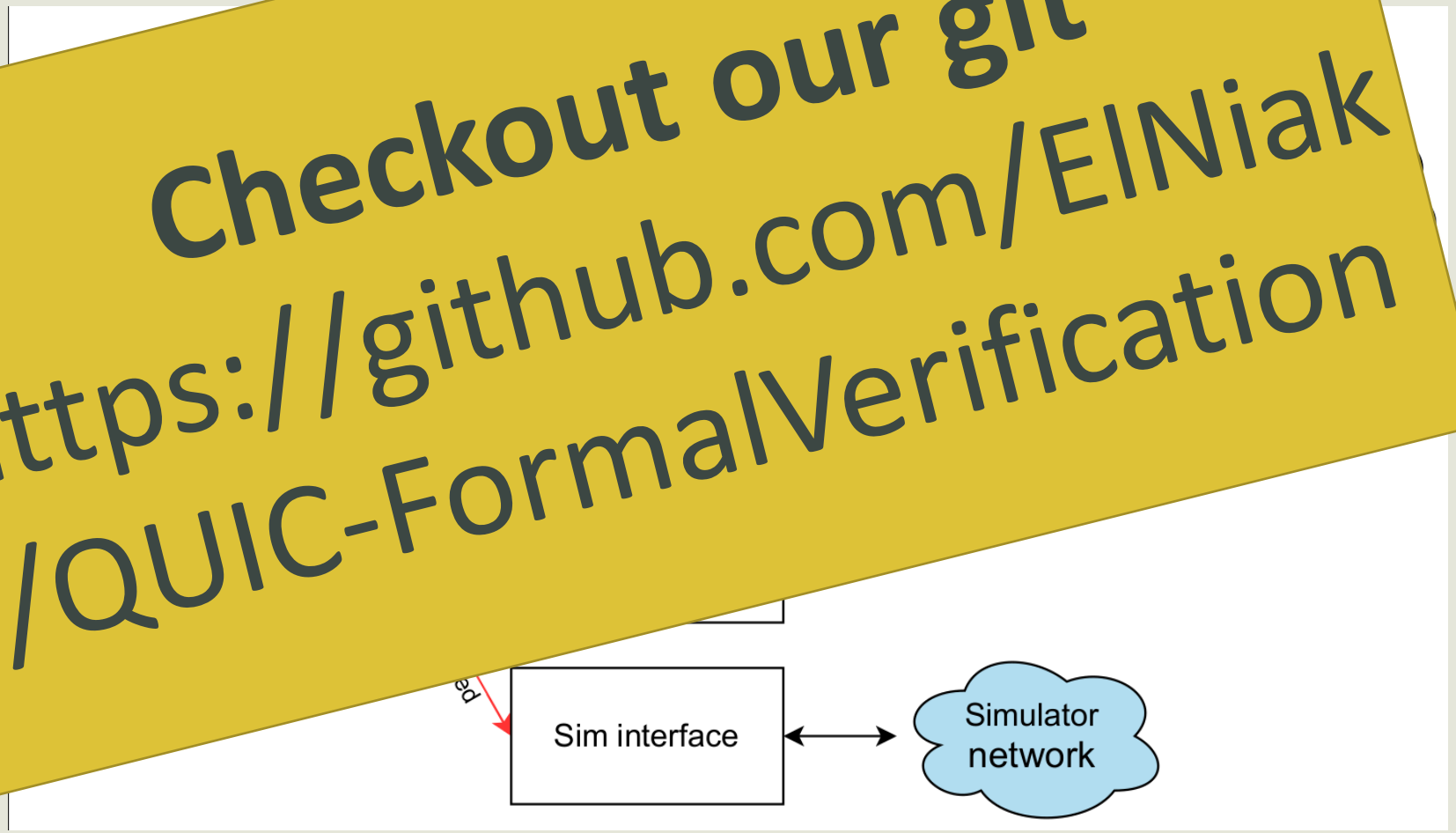
More realist experiments

+

Liveness property assumptions



Simulator



Checkout our git

https://github.com/ElNiak/QUIC-FormalVerification

What's next ?

- Develops more complex templates
- Extend the methodology to other protocol (i.e DNS)
- Improve the GUI for easier configuration

- Many more

UCLouvain

Any question ?

Thanks for your attention

ÉCOLE POLYTECHNIQUE DE LOUVAIN