Runtime Verification as a Toolkit of Techniques for Cyber Security Monitoring

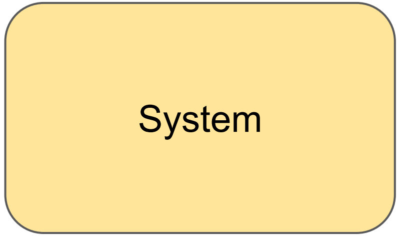
Christian Colombo

University of Malta

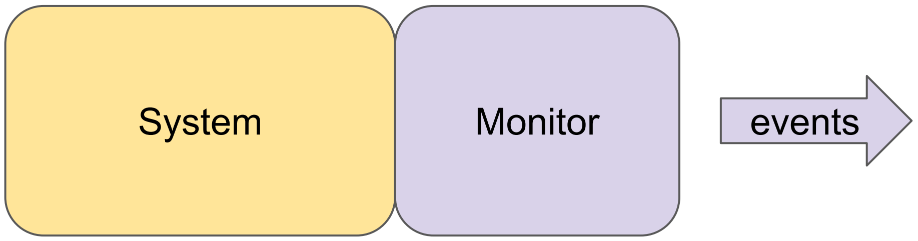
Genova, April 2023

What is Runtime Verification?

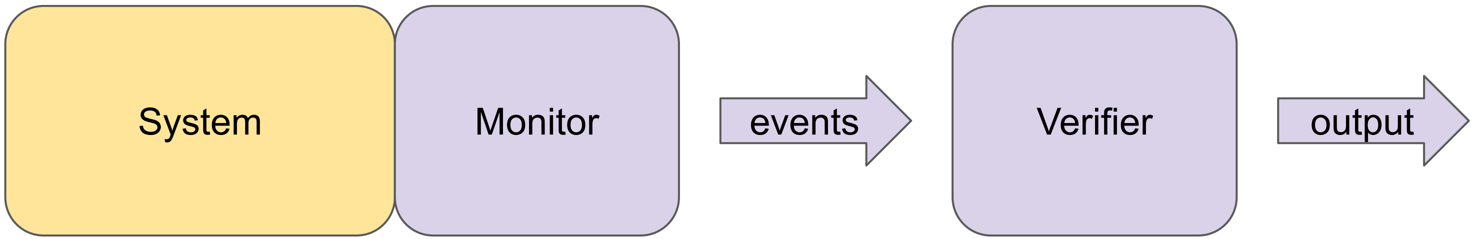
A General Picture



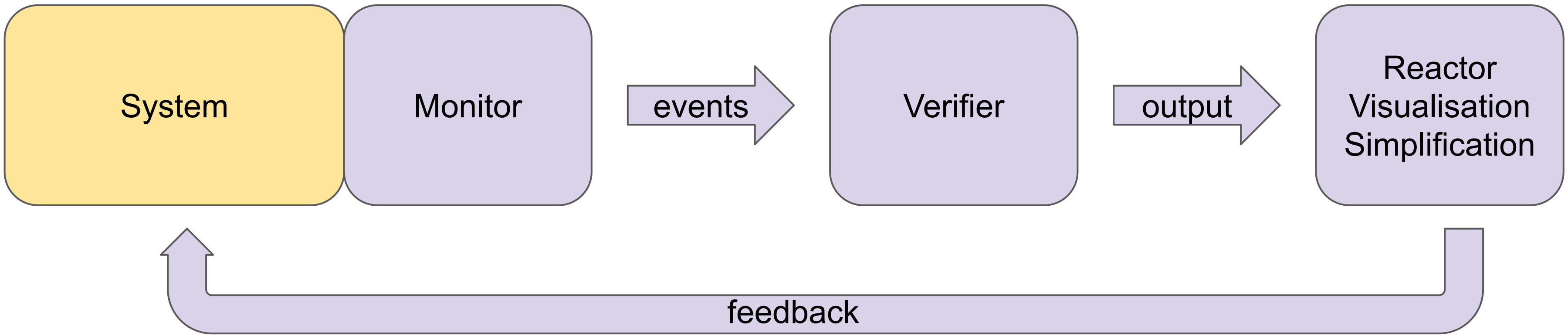
A General Picture



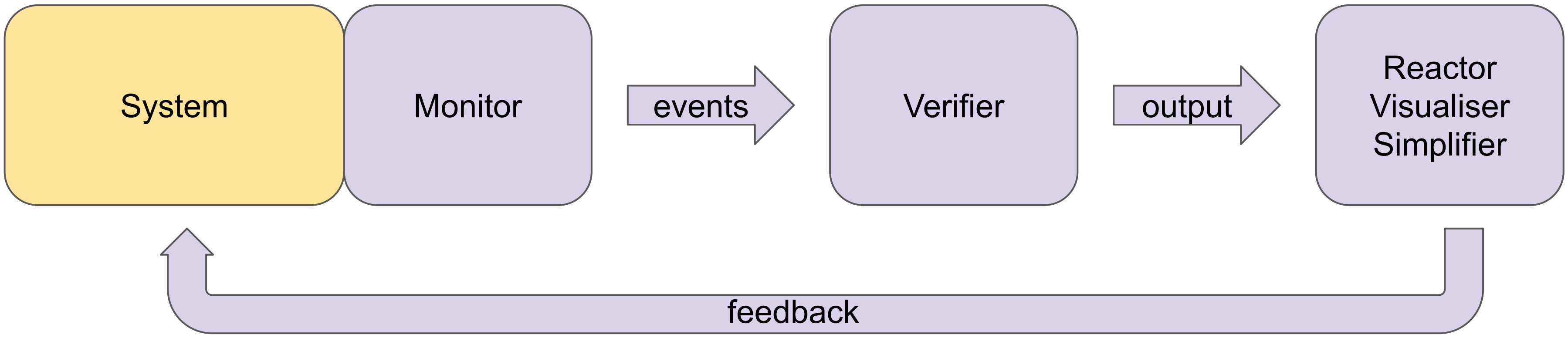
A General Picture



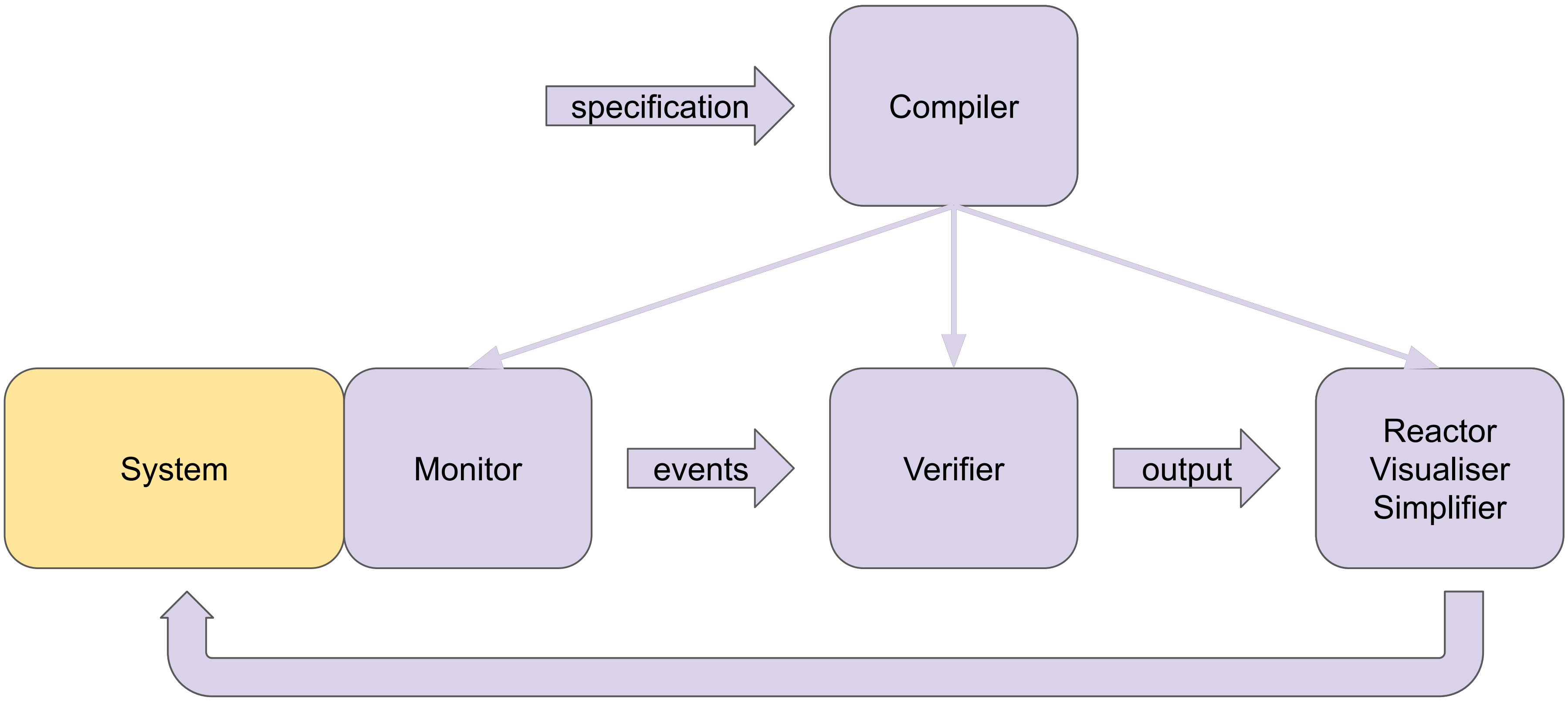
A General Picture



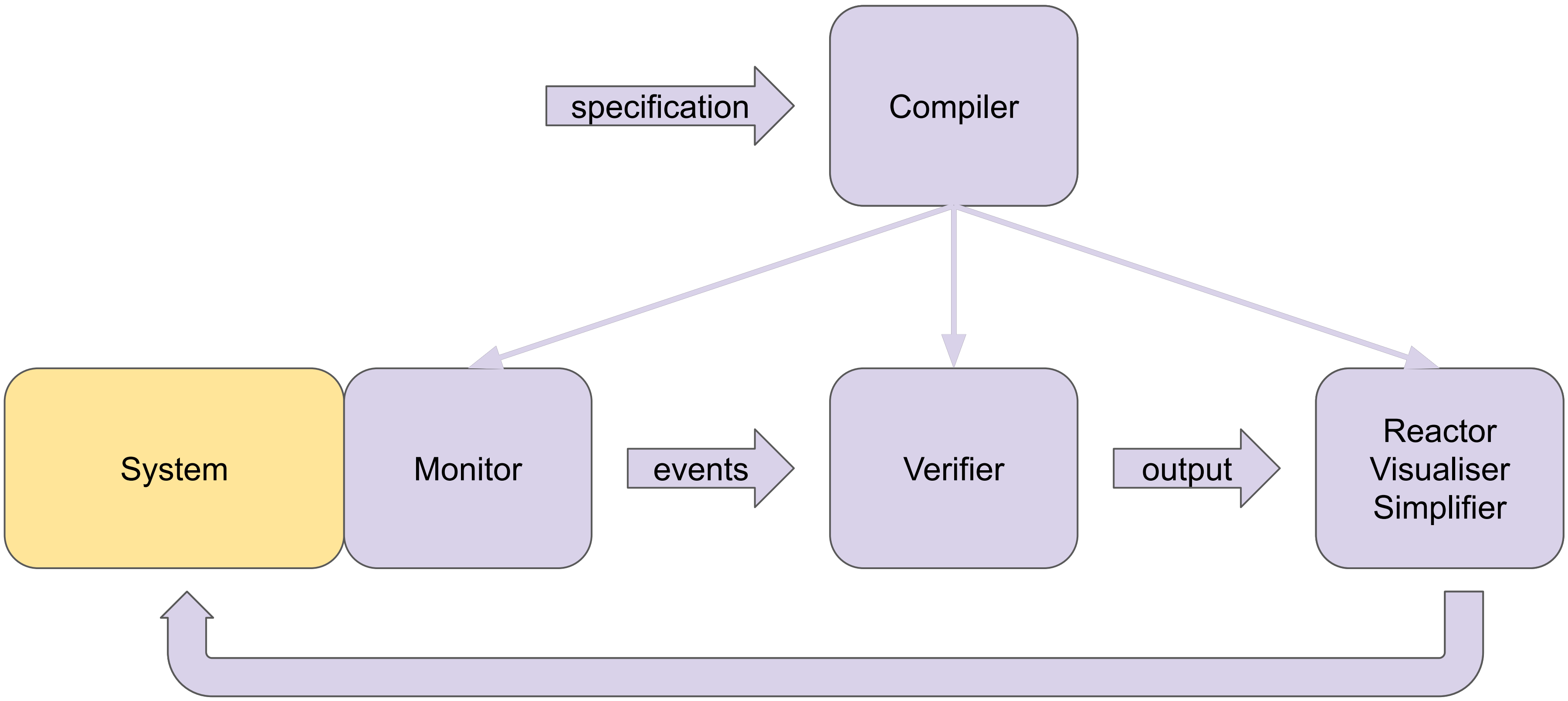
What is Special?



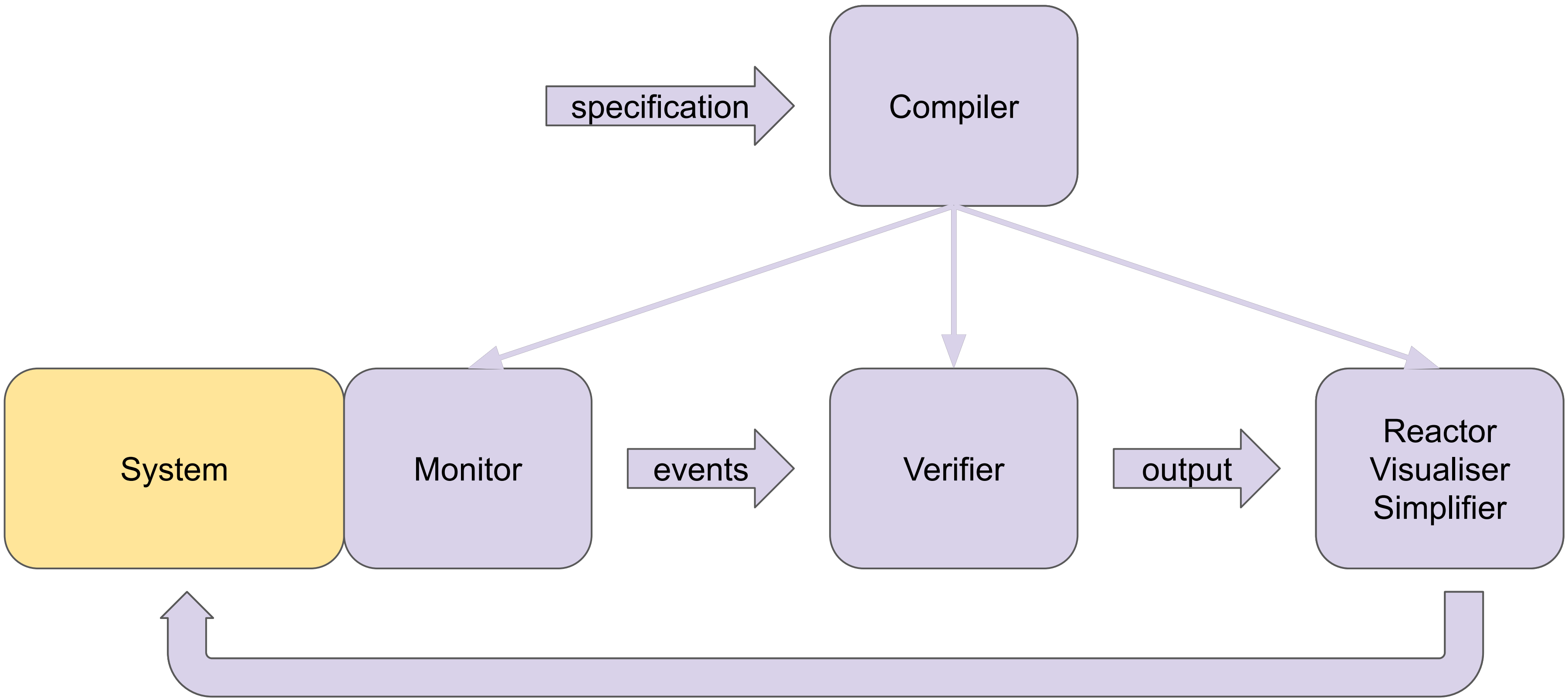
What is Special? (1) - Automatic Synthesis



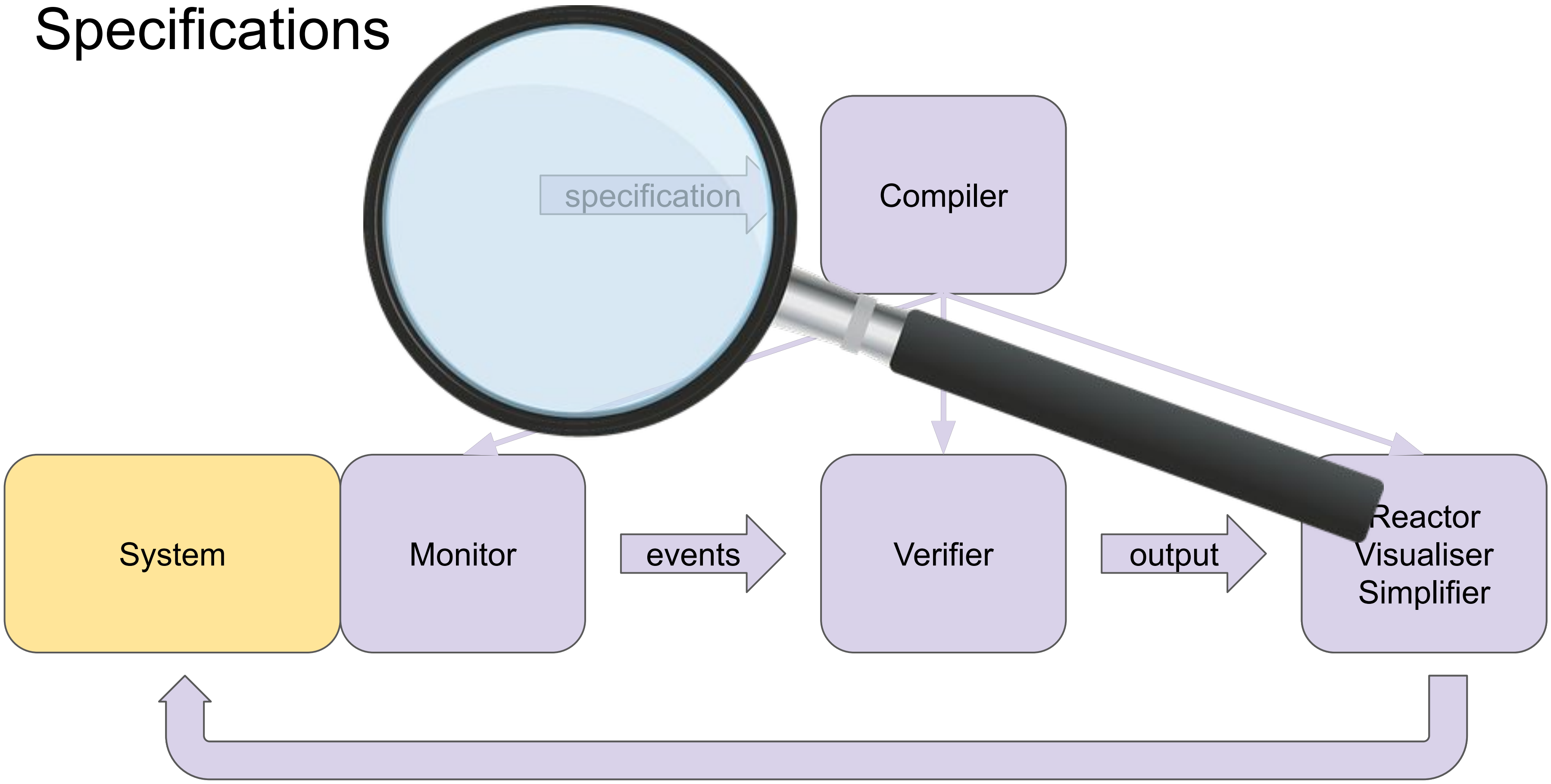
What is Special? (2) - Separation of Concerns



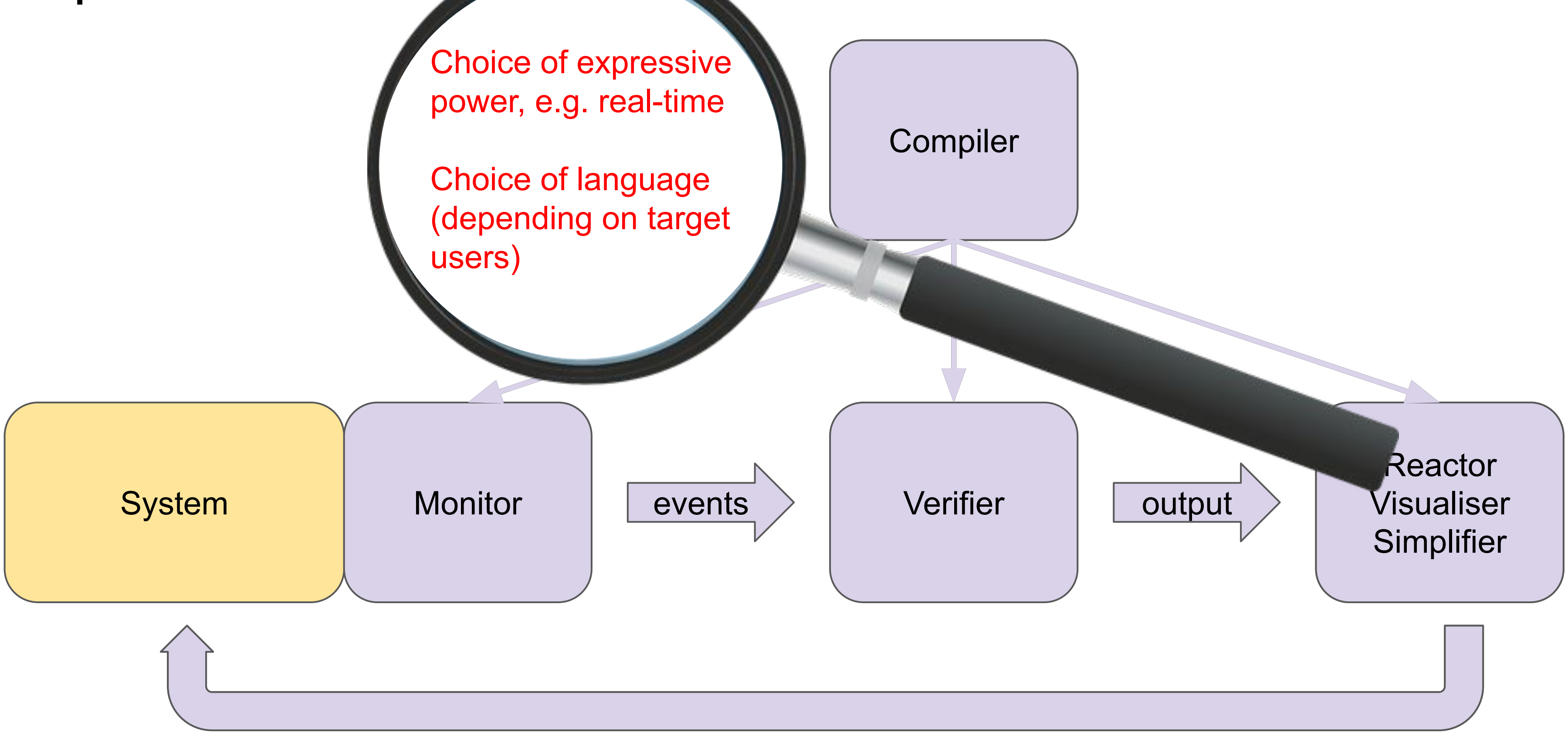
Zooming in


specification
Compiler
System
Monitor
events
Verifier
output
Reactor
Visualiser
Simplifier

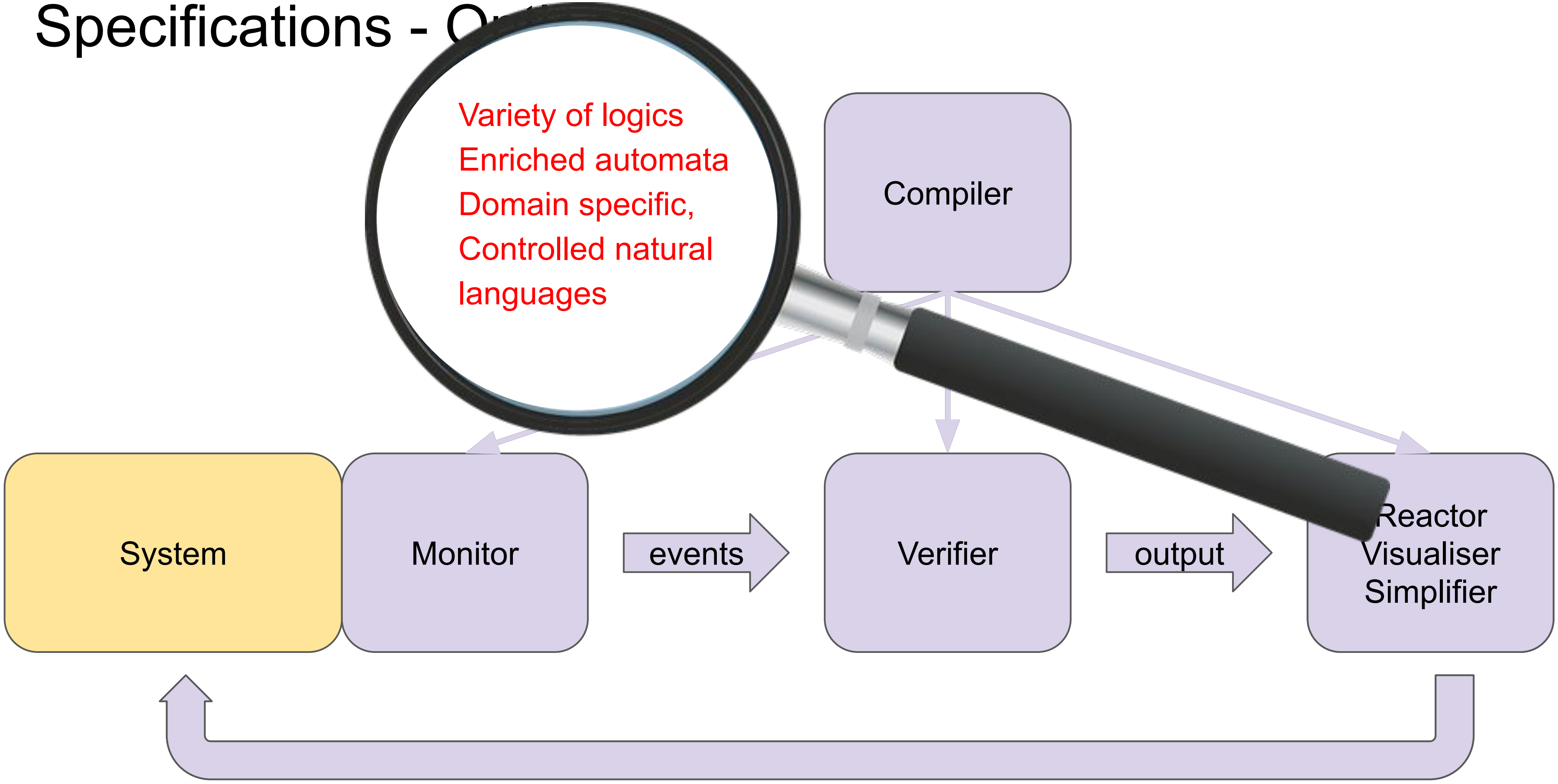
Specifications



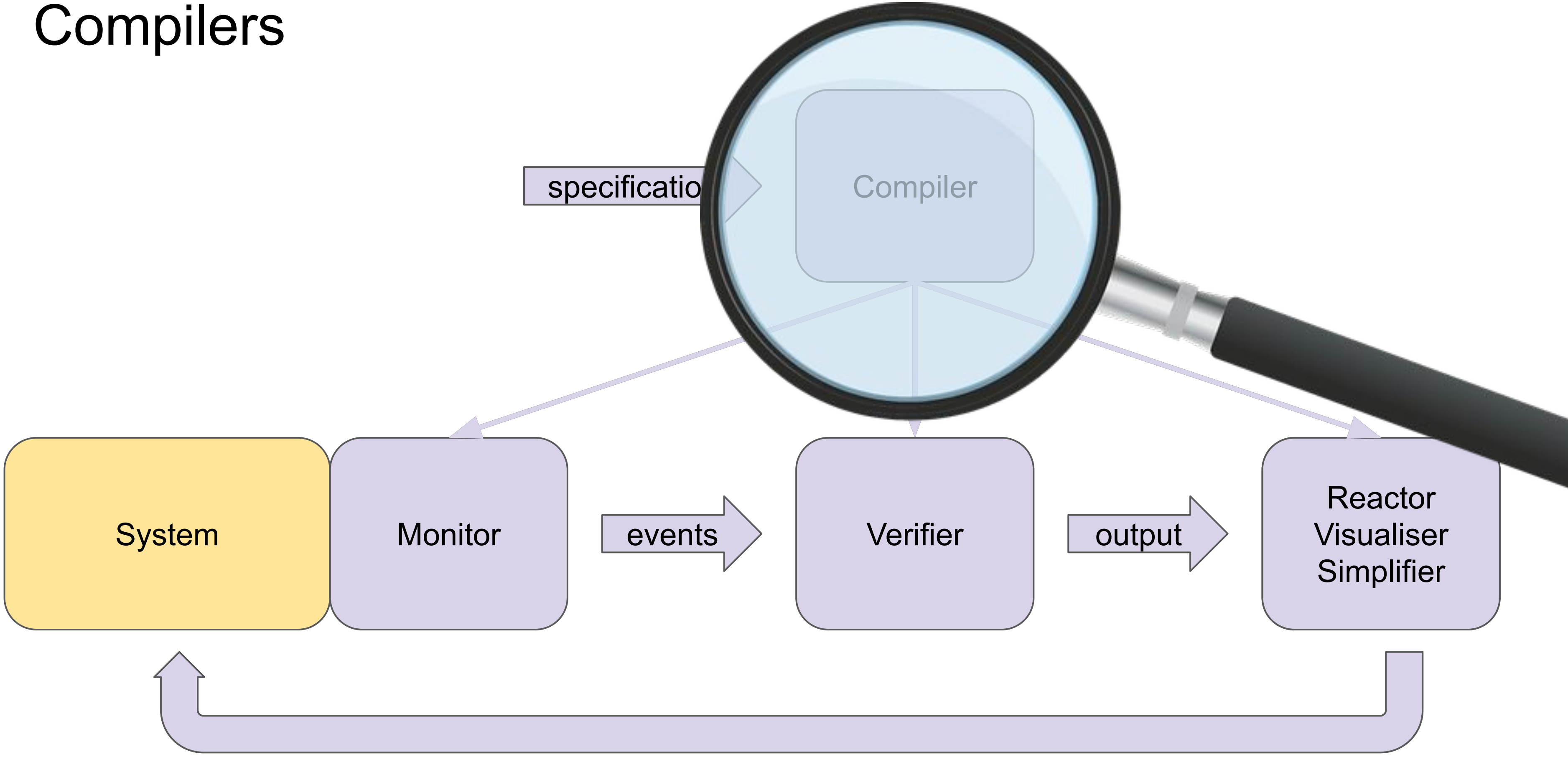
Specifications - Ch[illegible]es



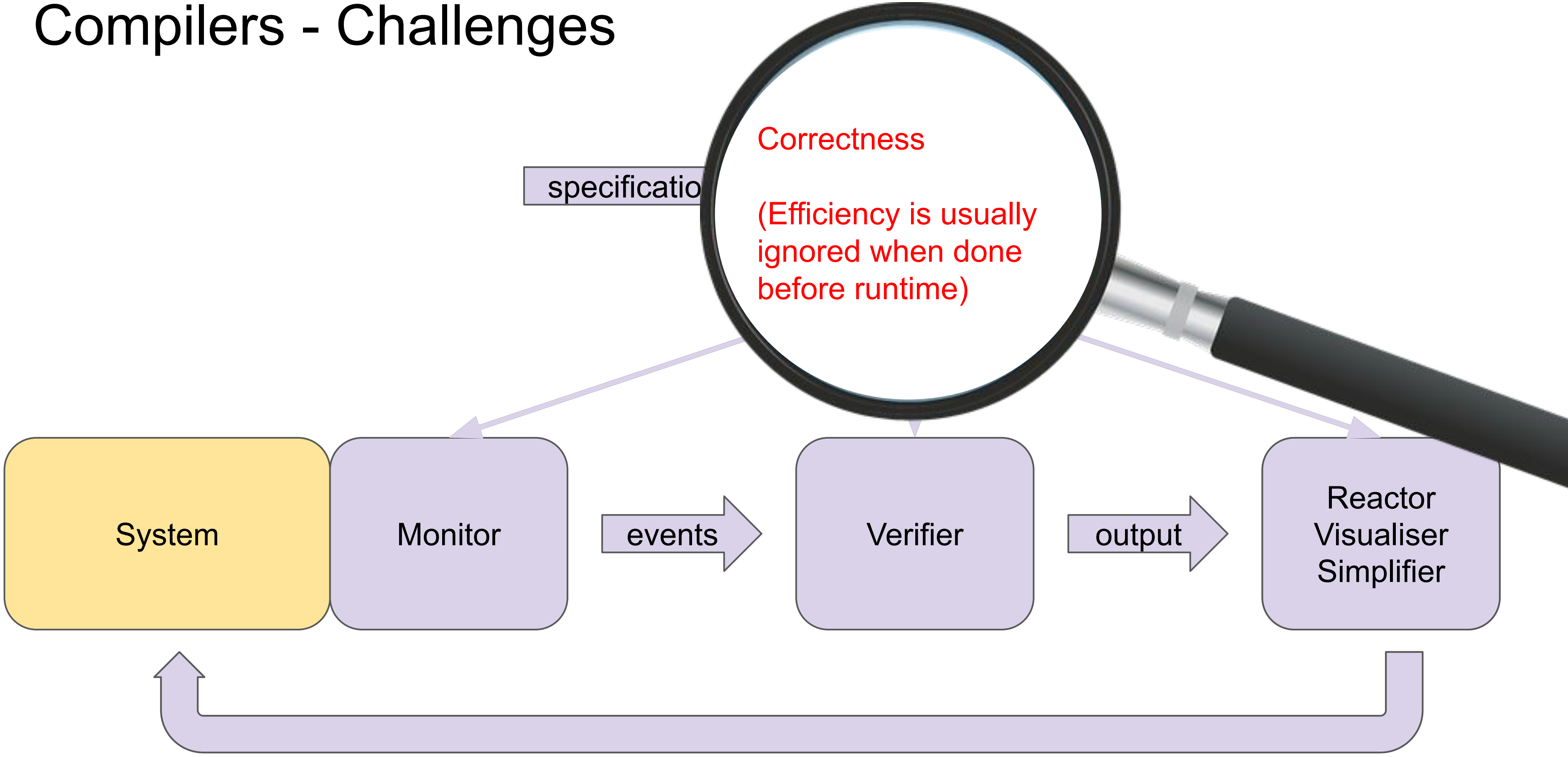
Specifications - O



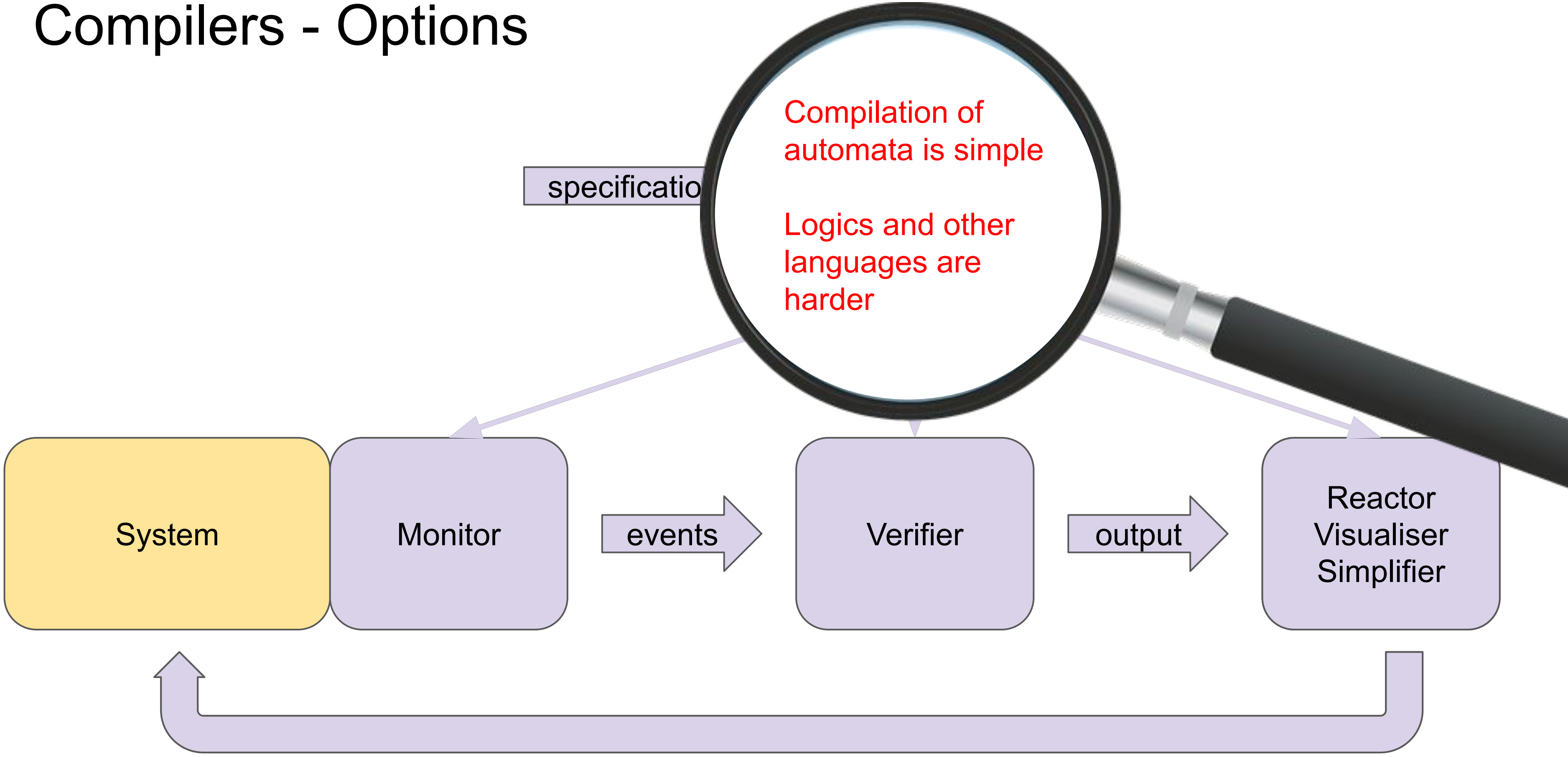
Compilers


specificatio
Compiler
System
Monitor
events
Verifier
output
Reactor
Visualiser
Simplifier

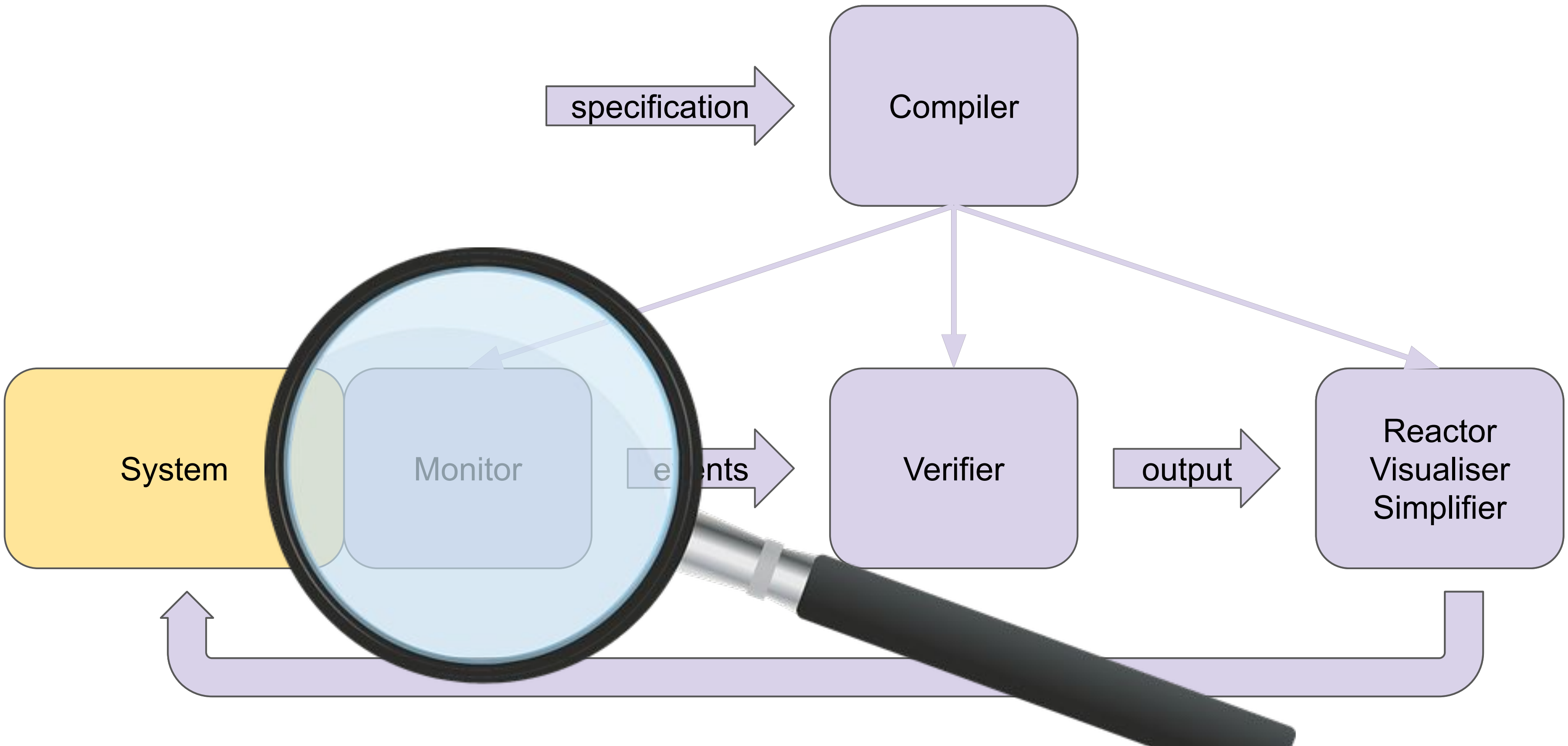
Compilers - Challenges



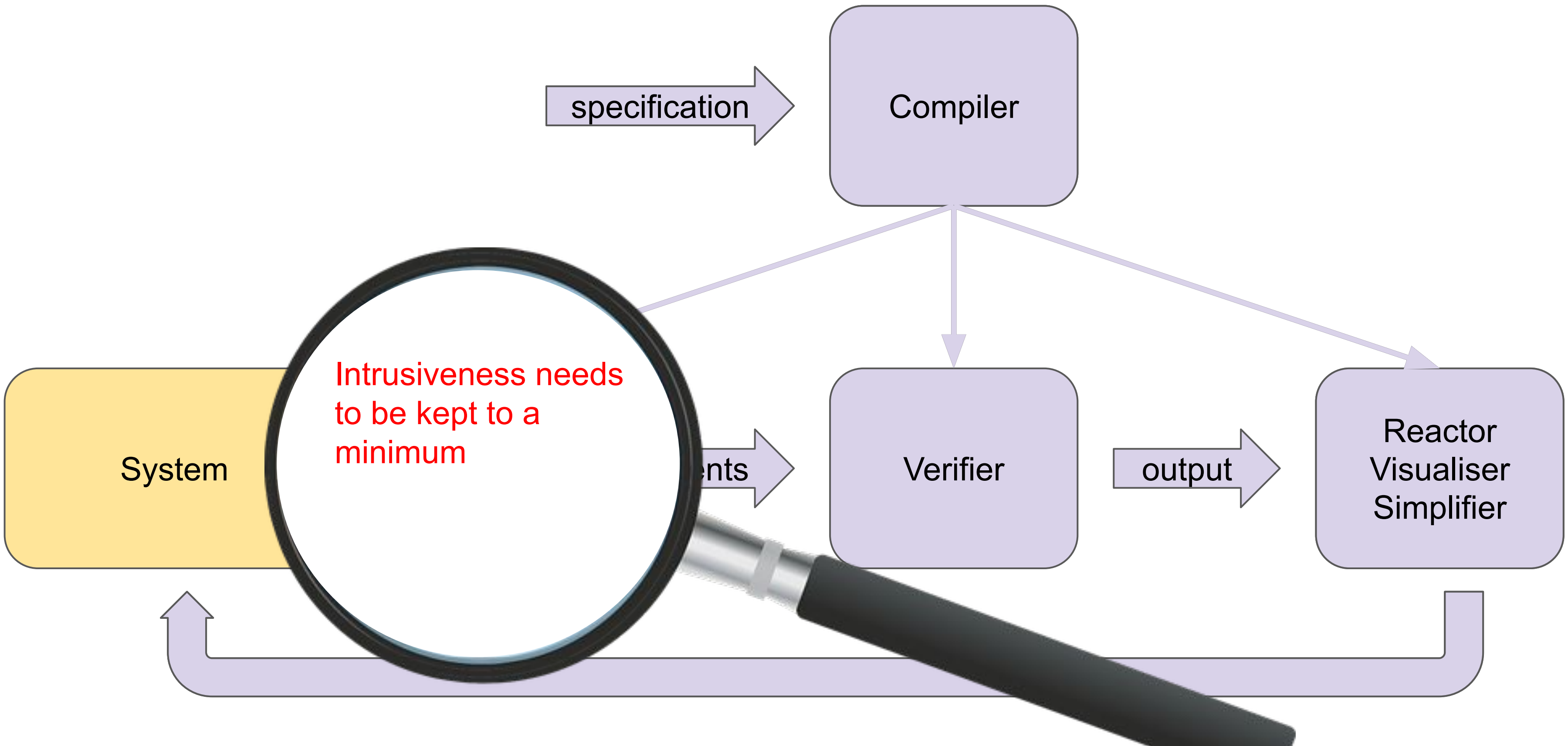
Compilers - Options



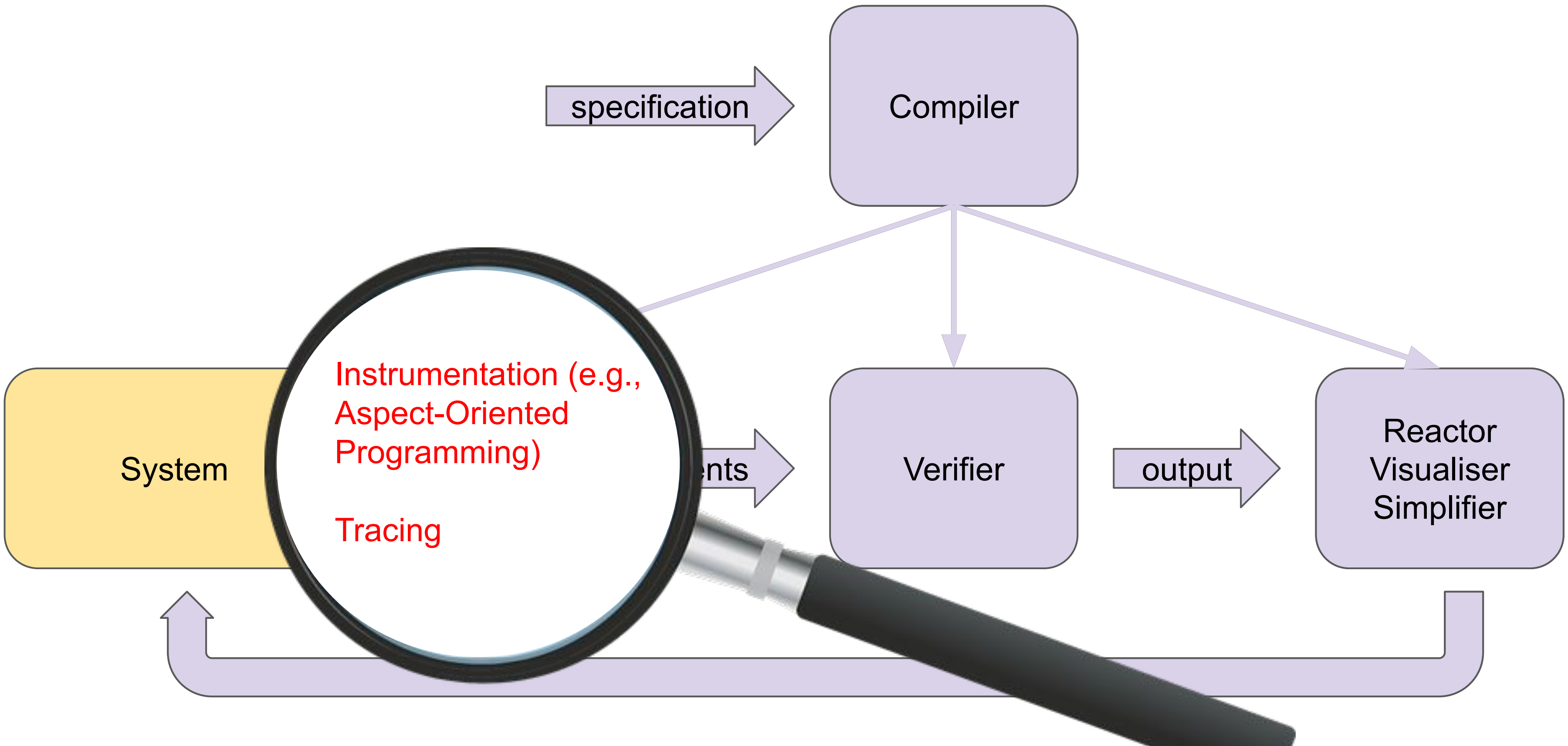
Monitors


specification
Compiler
System
Monitor
events
Verifier
output
Reactor
Visualiser
Simplifier

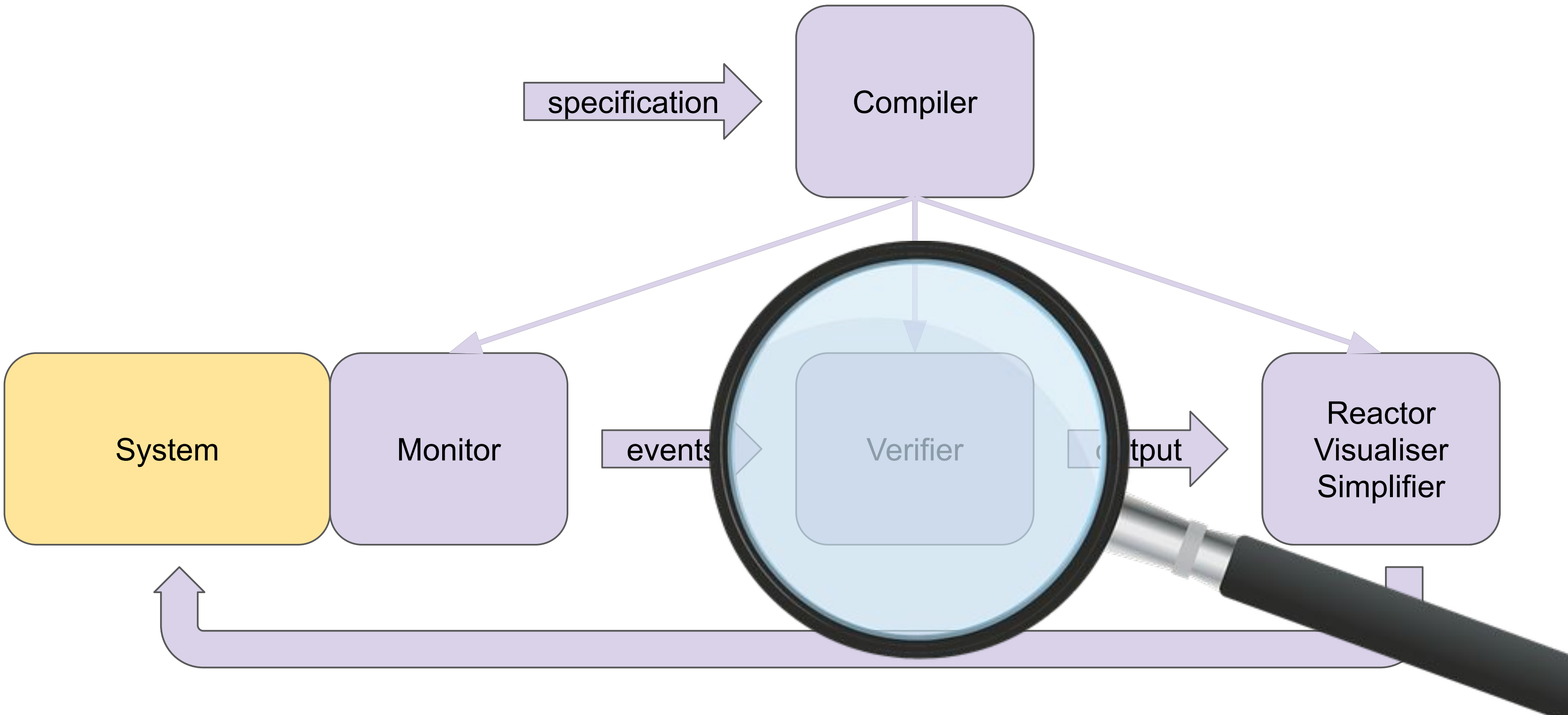
Monitors - Challenges



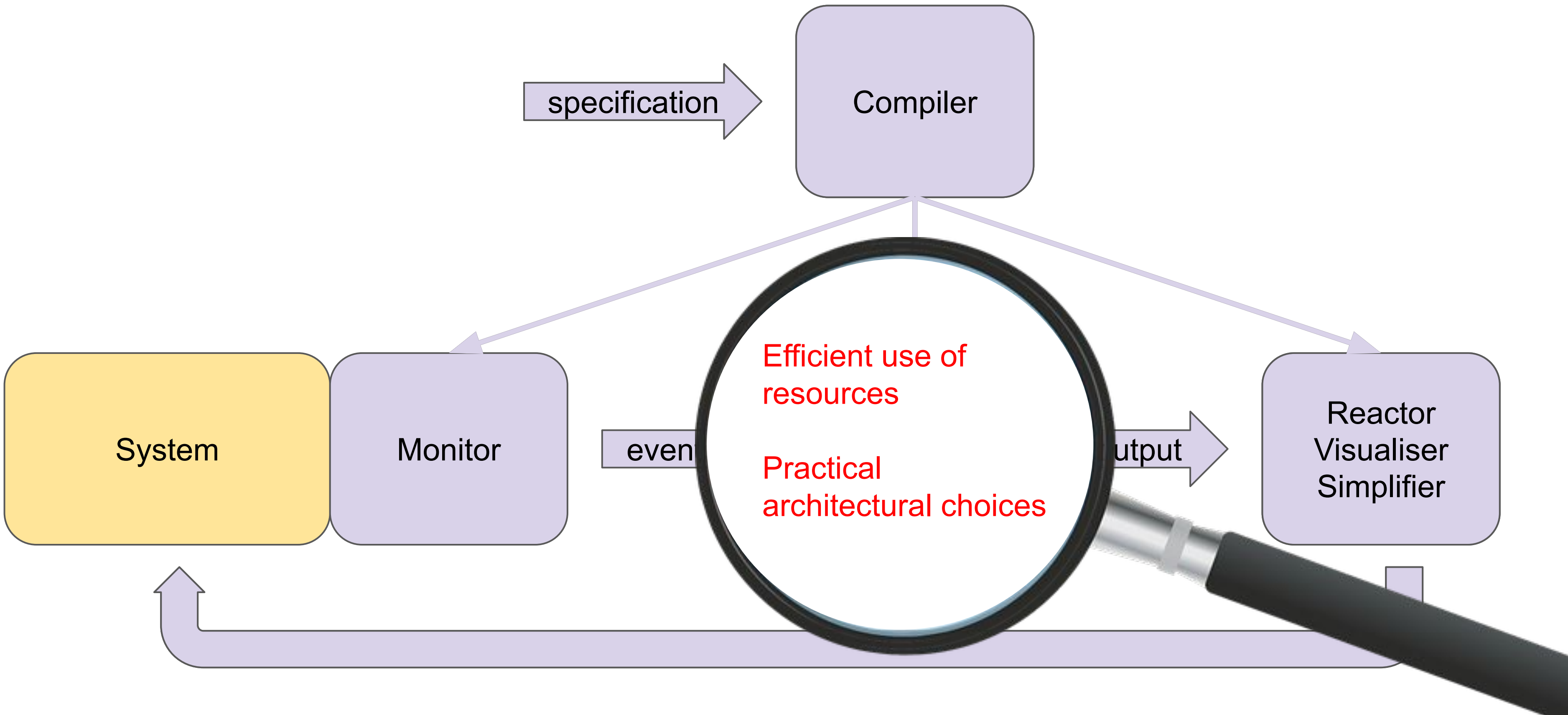
Monitors - Options



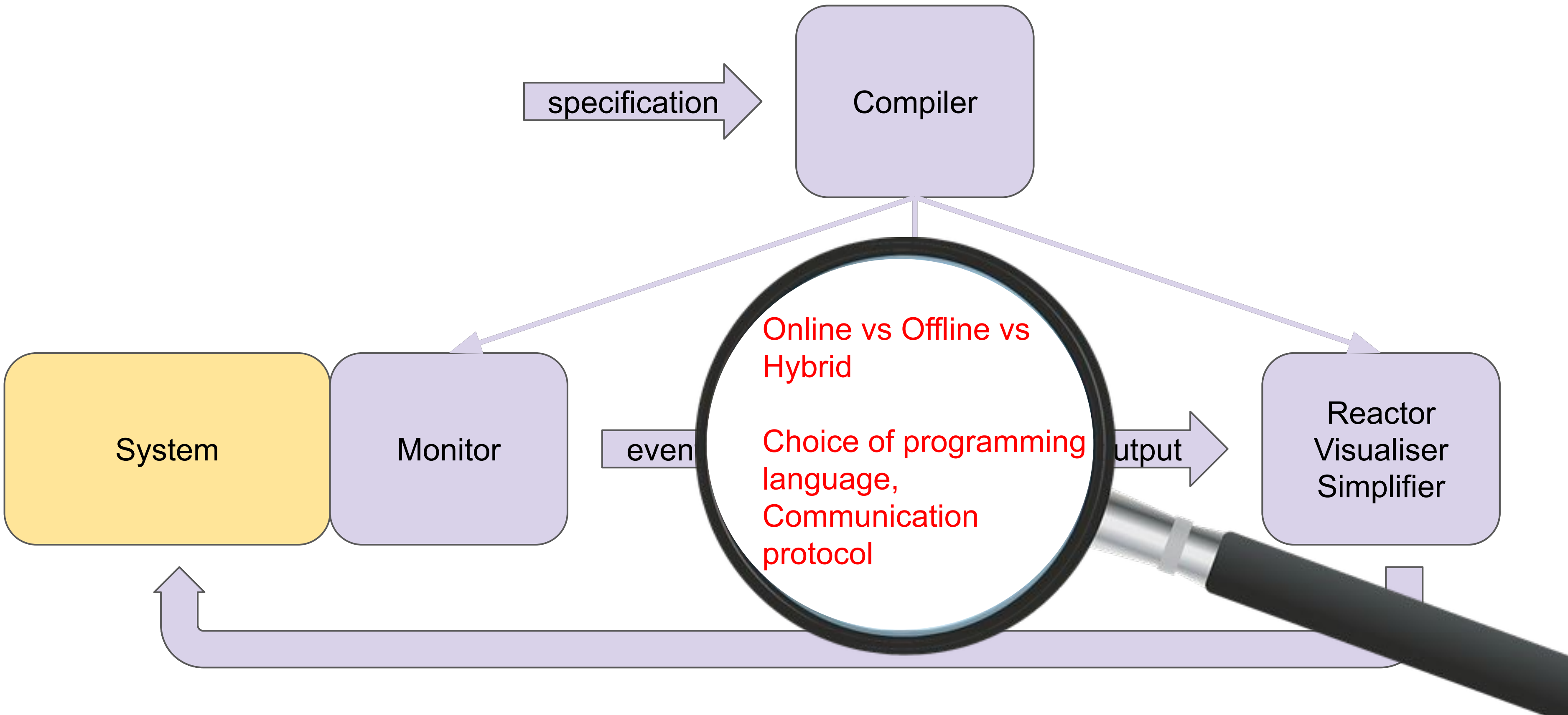
Verifiers


specification
Compiler
System
Monitor
events
Verifier
Reactor
Visualiser
Simplifier

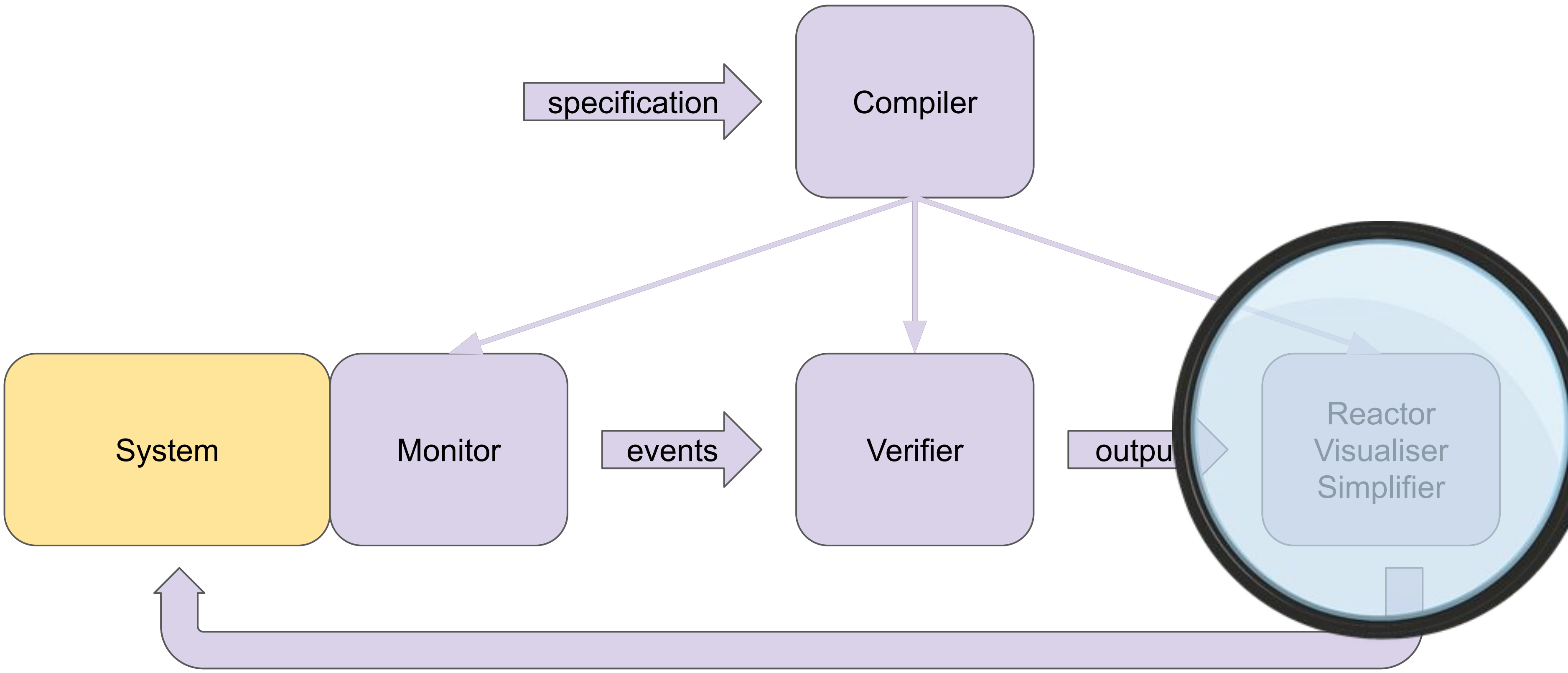
Verifiers - Challenges



Verifiers - Options

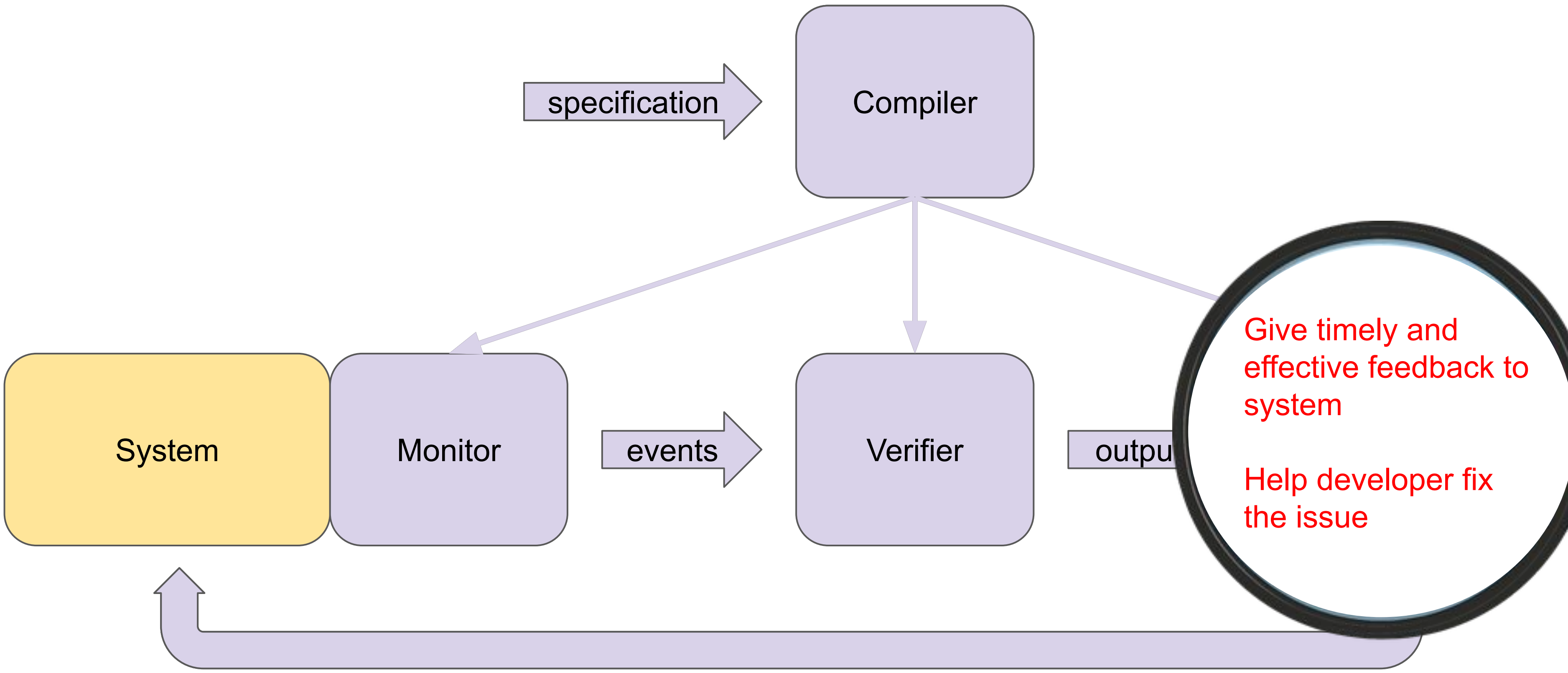


Reactions

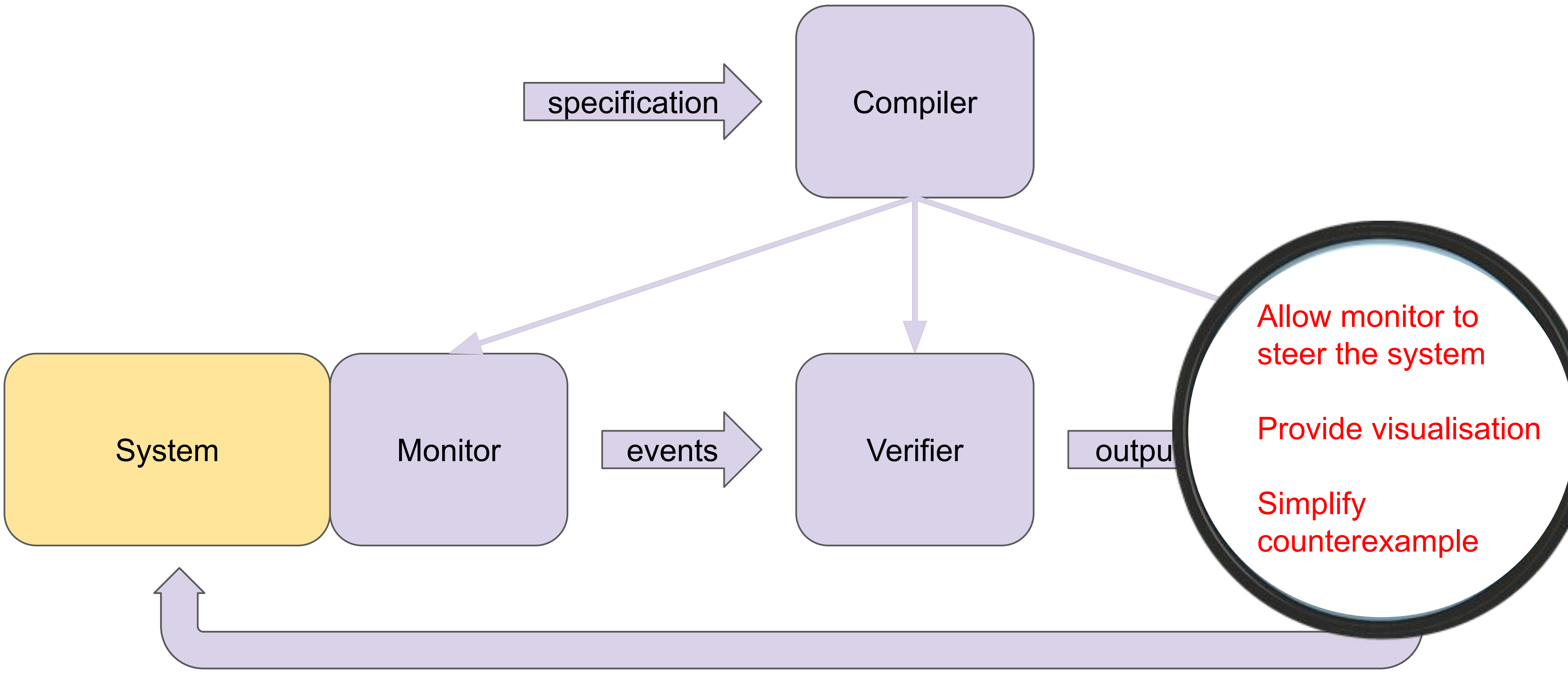


specification
Compiler
System
Monitor
events
Verifier
outpu
Reactor
Visualiser
Simplifier

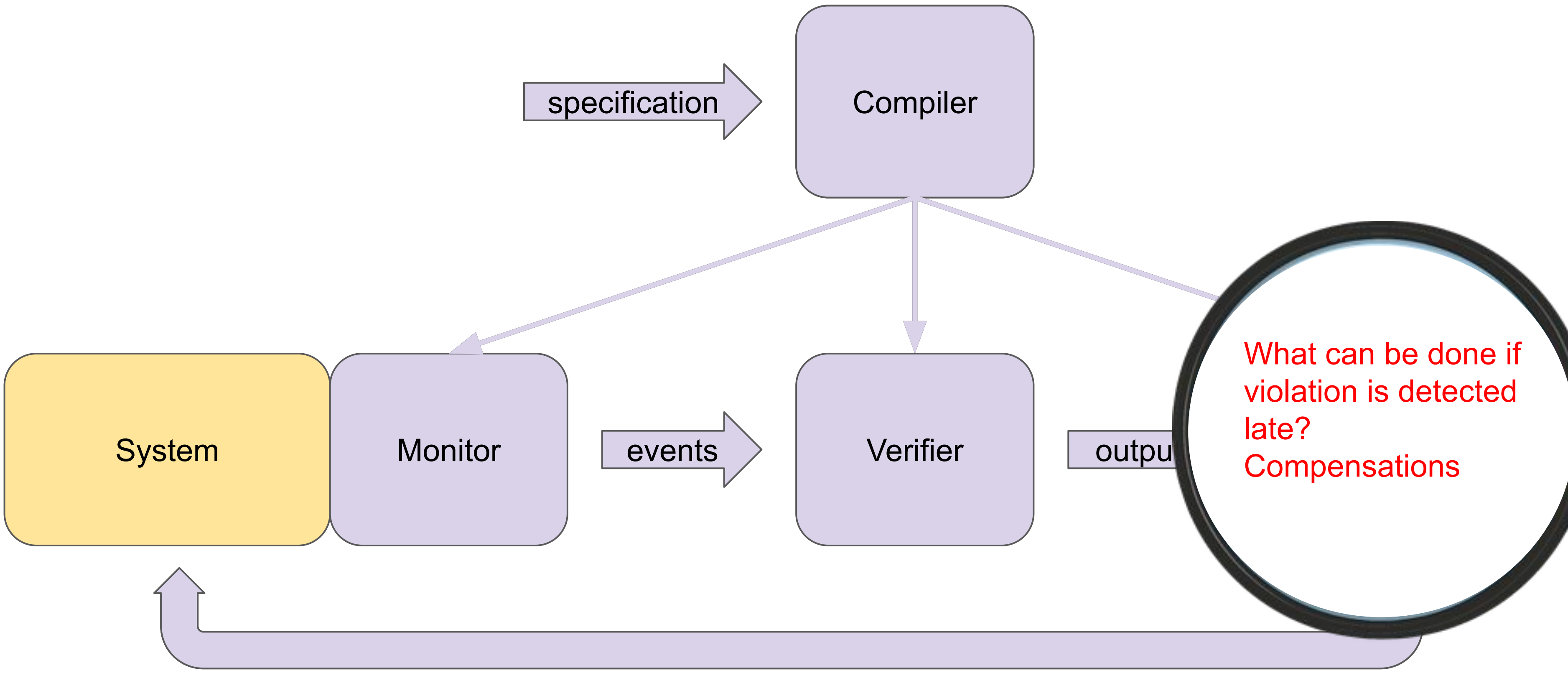
Reactions - Challenges



Reactions - Options



Reactions - More Options



Cyber Security (1)
Securing a Group Key Exchange Protocol Implementation

(part of NATO-funded project)

Secure Communication in the Quantum Era

Quantum computers (when they become practical) pose a threat to cryptographic communication protocols.

This project aimed to design a new “Quantum-safe” Group Key Exchange Protocol

And provide a proof of concept implementation

Case study: a chat application using the protocol to establish the secret keys

L-Università ta' Malta

NATO OTAN

The need for secure communication

As the COVID-19 pandemic lockdown forced most employees into remote working, serious weaknesses in Zoom were exposed. Issues ranged from insecure key establishment to inadequate block cipher mode usage.

Other previous high-profile incidents concerning insecure cryptographic protocol implementation were caused by:

- Weak randomness.
- Insufficient checks on protocol compliance.
- Memory corruption bugs.

Many things can go wrong on many different levels of abstraction

(High level) Wrong protocol implementation	The protocol implementation might deviate from the verified (theoretical) design
Medium level threats	Malware, Data leaks, etc
Low level threats	Arithmetic overflows, undefined downcasts, and invalid pointer references
Hardware	Can hardware be trusted? Side Channel attacks?

L-Università ta' Malta

NATO OTAN

Many things can go wrong on many different levels of abstraction

(High level) Wrong protocol implementation
The protocol implementation might deviate from the verified (the

Medium level threats
Malw

Low level threats
Arithmetic
d downcasts, and invalid pointer references

Hardware
Can hardware be trusted?
Side Channel attacks?

How can RV help?

L-Università ta' Malta

NATO OTAN

Many things can go wrong on many different levels of abstraction

(High level) Wrong protocol implementation	The protocol implementation might deviate from the verified (the…
Medium level threats	Malw…
Low level threats	Arithm… …d downcasts, and invali…
Hardwar…	Side Cha…

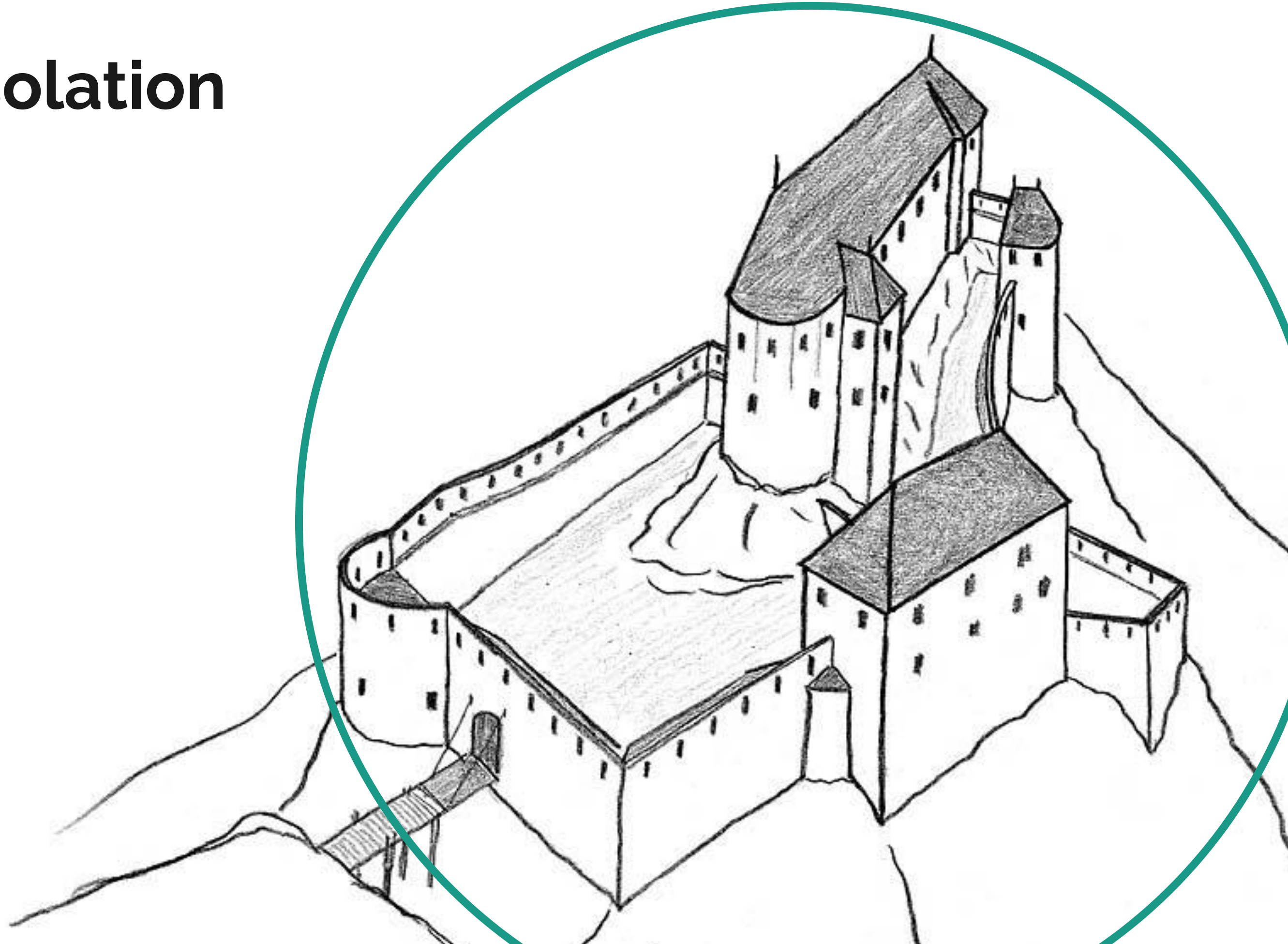
How can RV help?

A mix of RV and Hardware

L-Università ta' Malta

NATO OTAN

Concept 1: Isolation



Concept 1: Isolation

Medium level

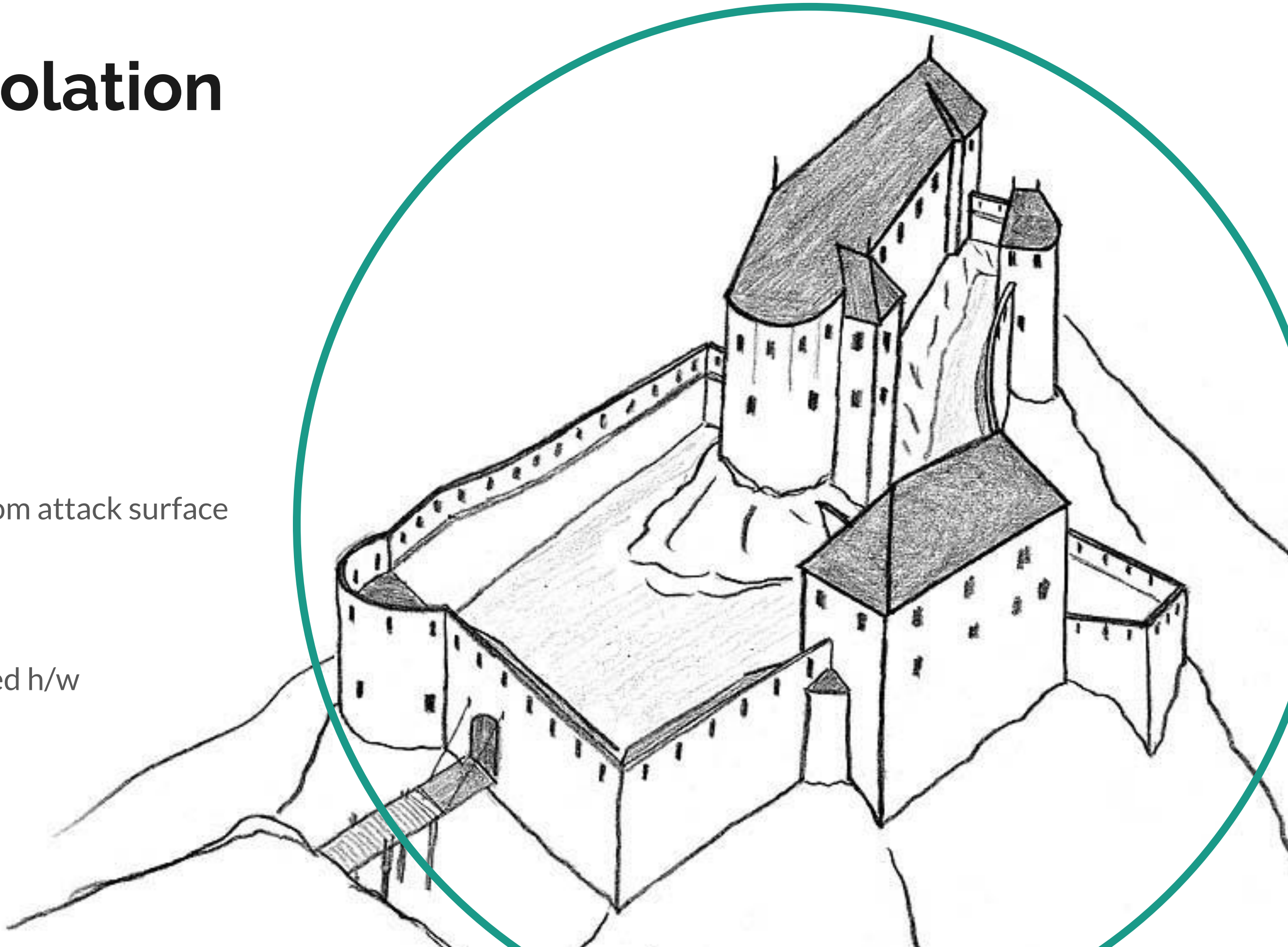
- Crypto keys

Low level

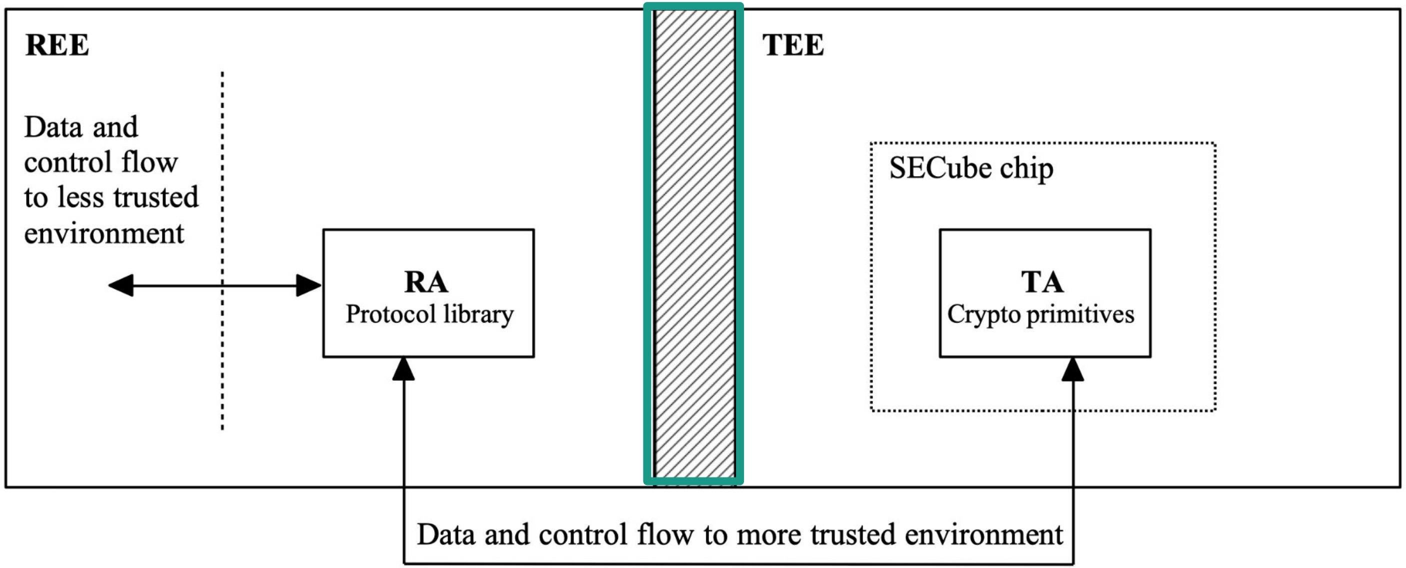
- Hide memory errors from attack surface

Hardware

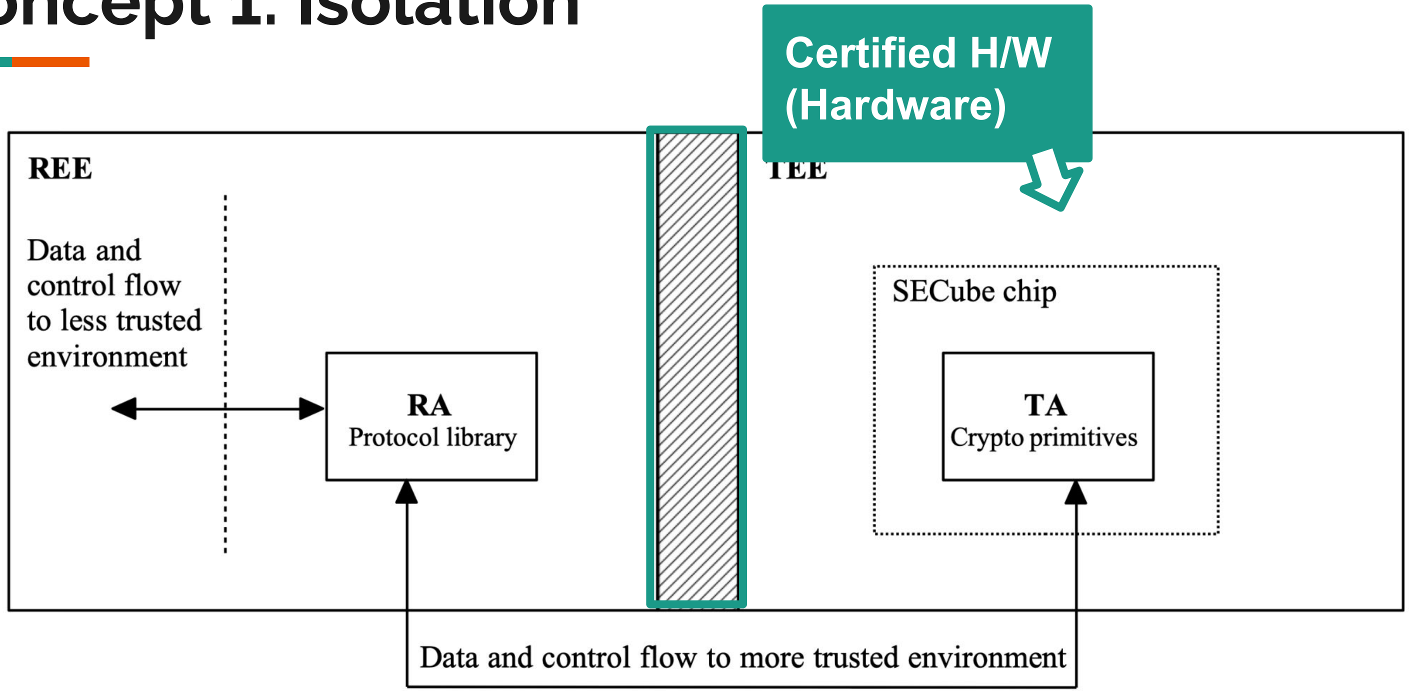
- Bring-your-own certified h/w



Concept 1: Isolation



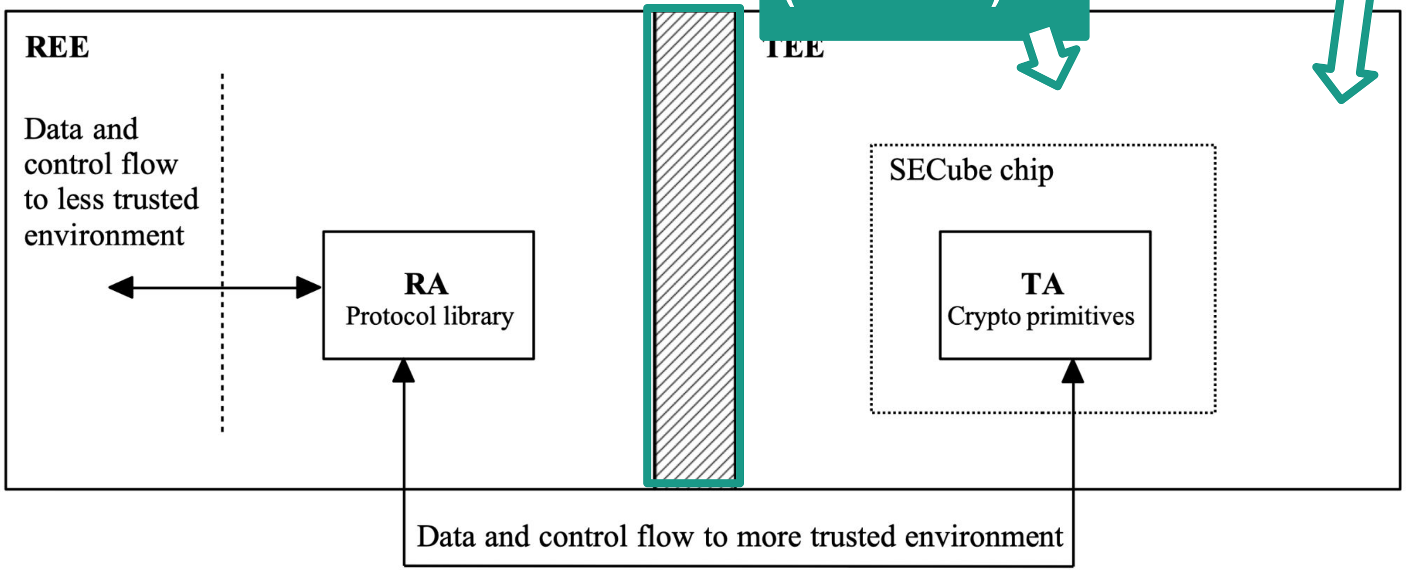
Concept 1: Isolation



Concept 1: Isolation

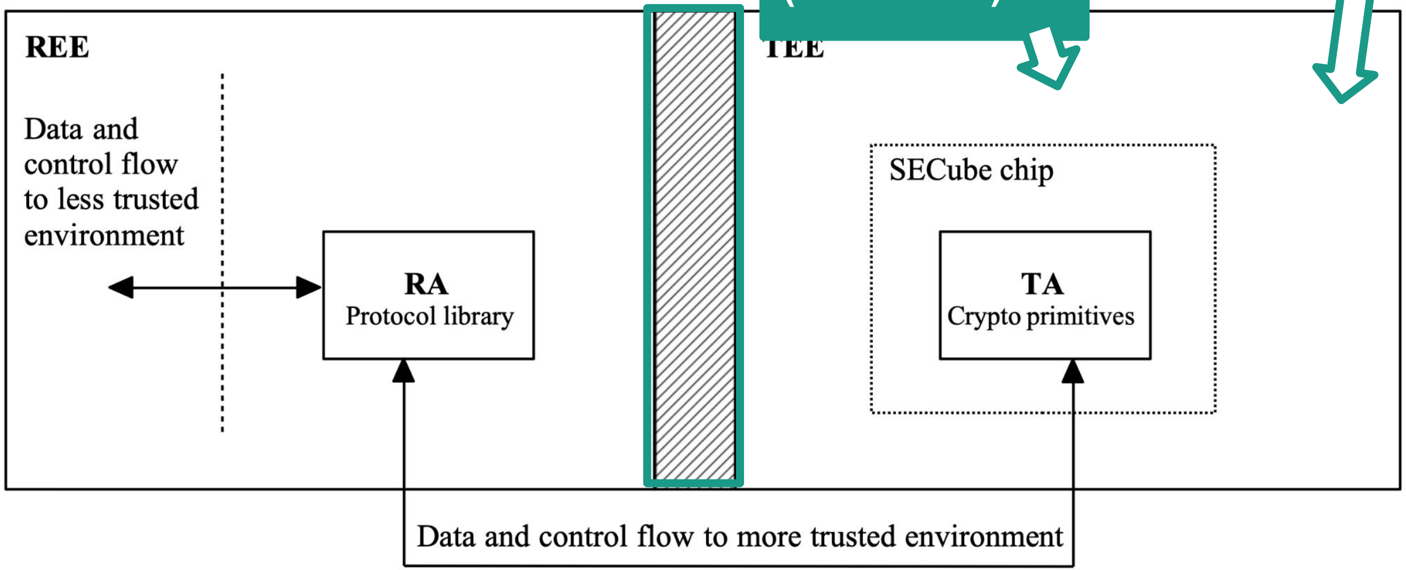
Certified H/W (Hardware)

Hide memory errors from attack surface (Low level)



L-Università ta' Malta

Concept 1: Isolation

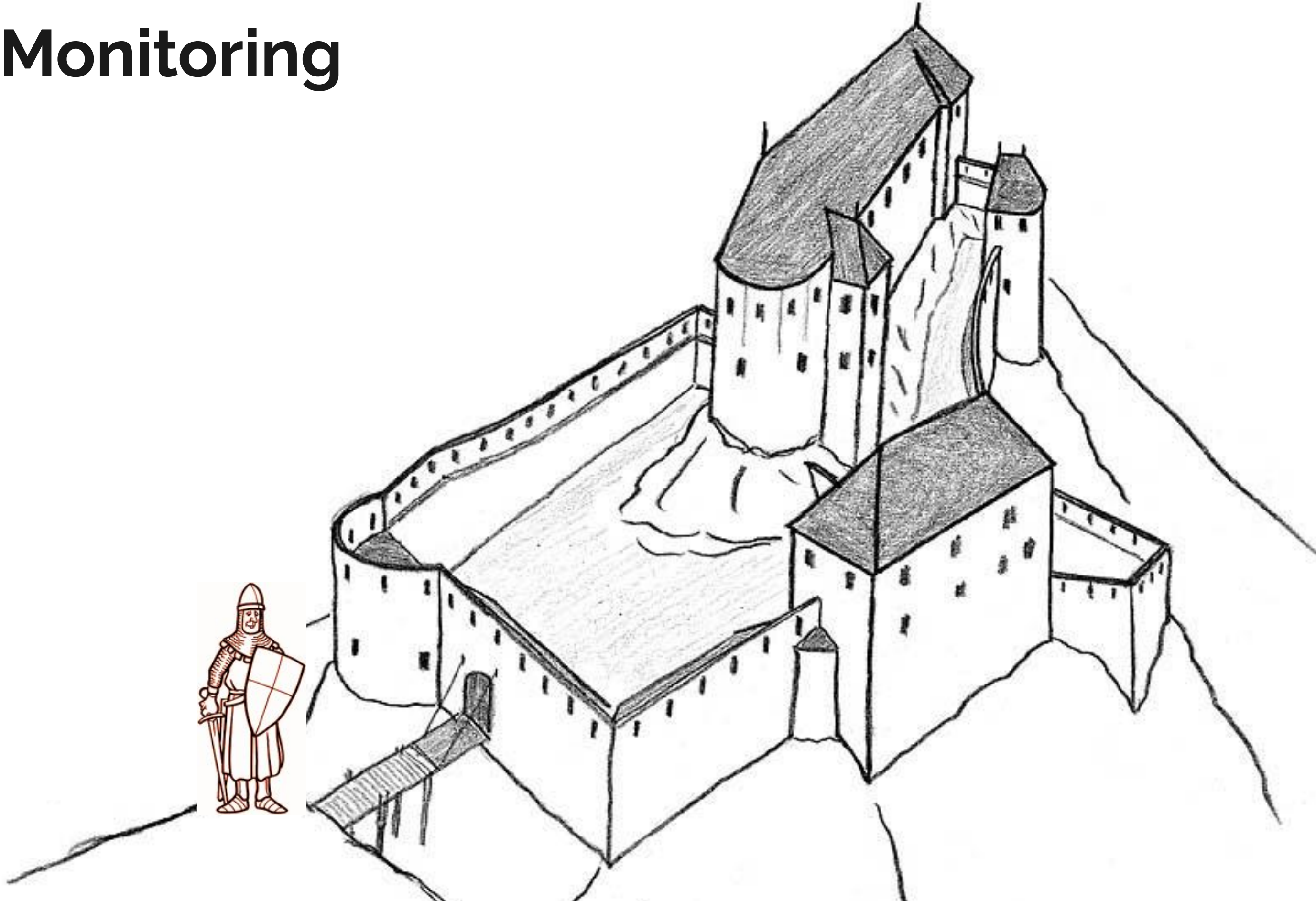


Certified H/W (Hardware)

Hide memory errors from attack surface (Low level)

No crypto key transfers (Medium level)

Concept 2: Monitoring



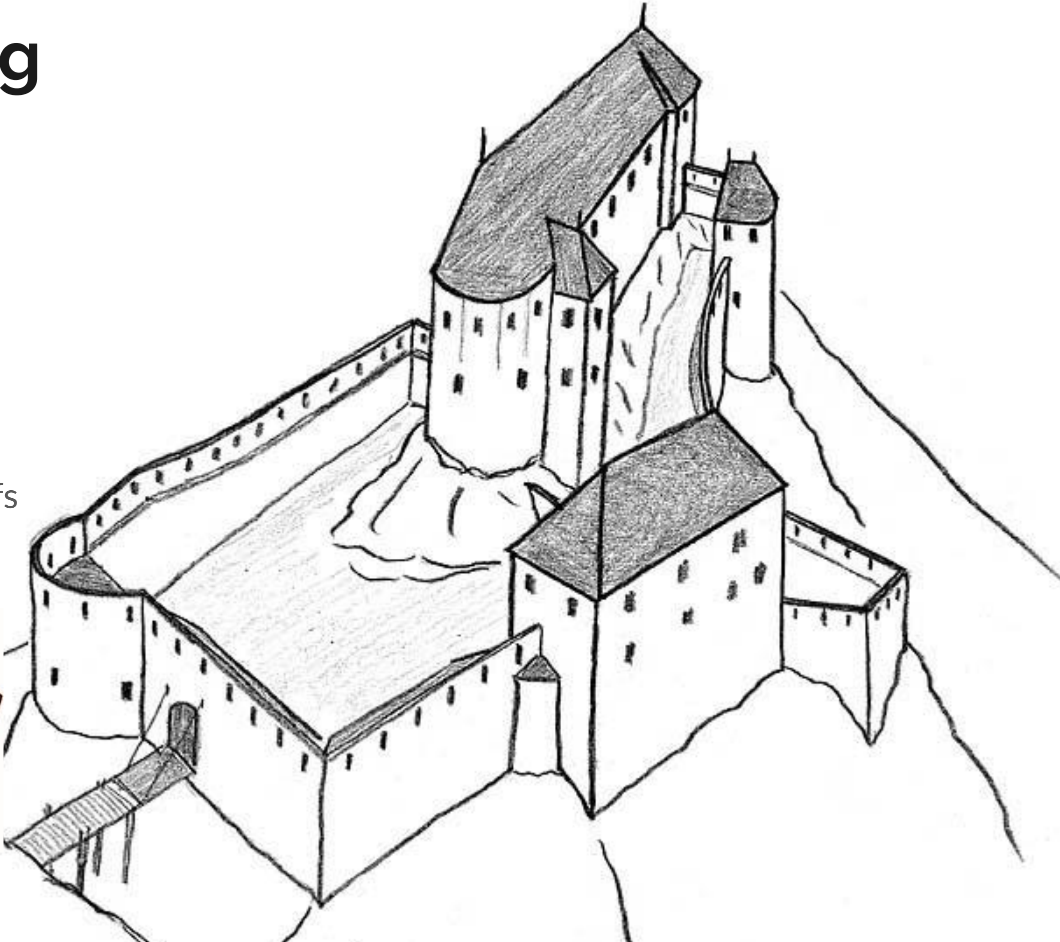
Concept 2: Monitoring

High level

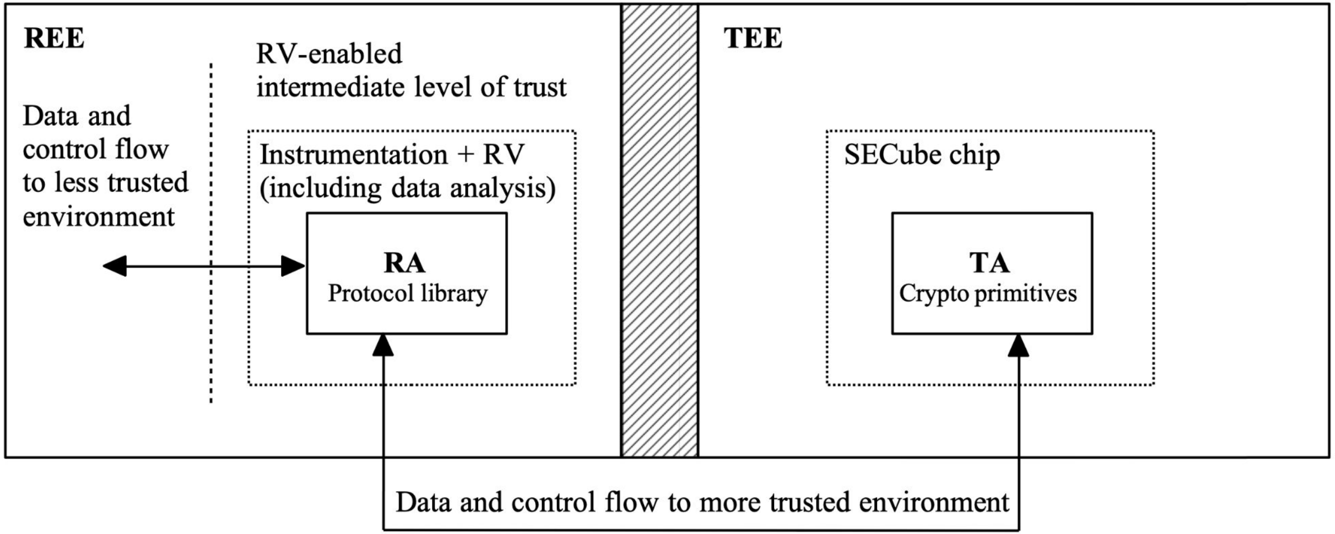
- Check sequences of actions
- Check parameters
- Check any assumptions implicit in proofs

Low level

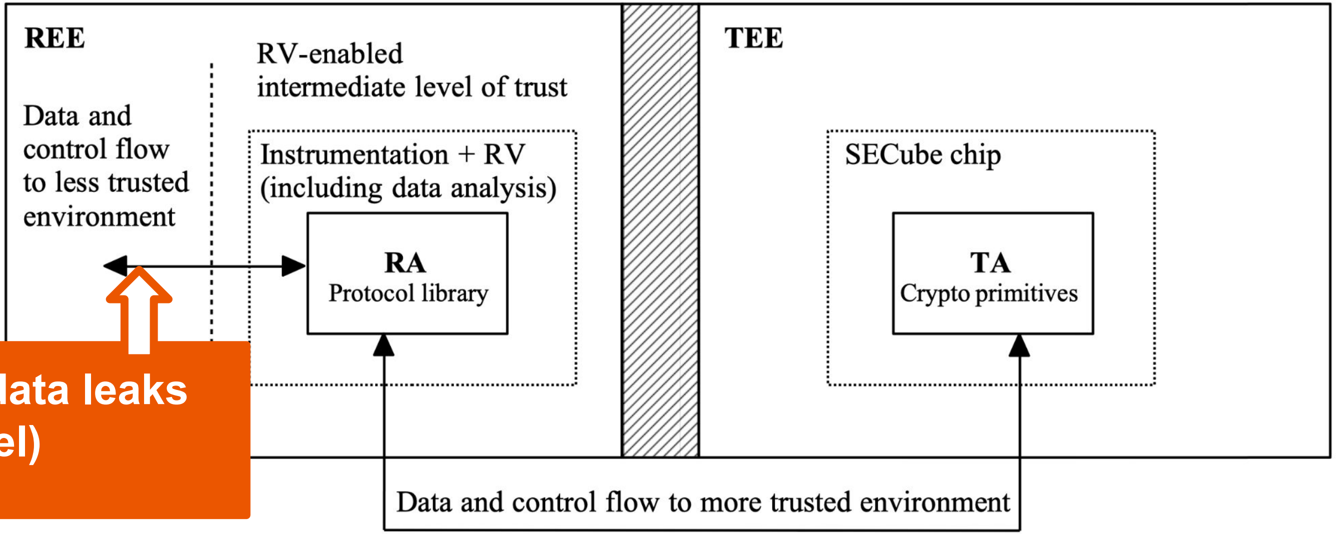
- Plaintext leakage across border



Concept 2: Monitoring



Concept 2: Monitoring

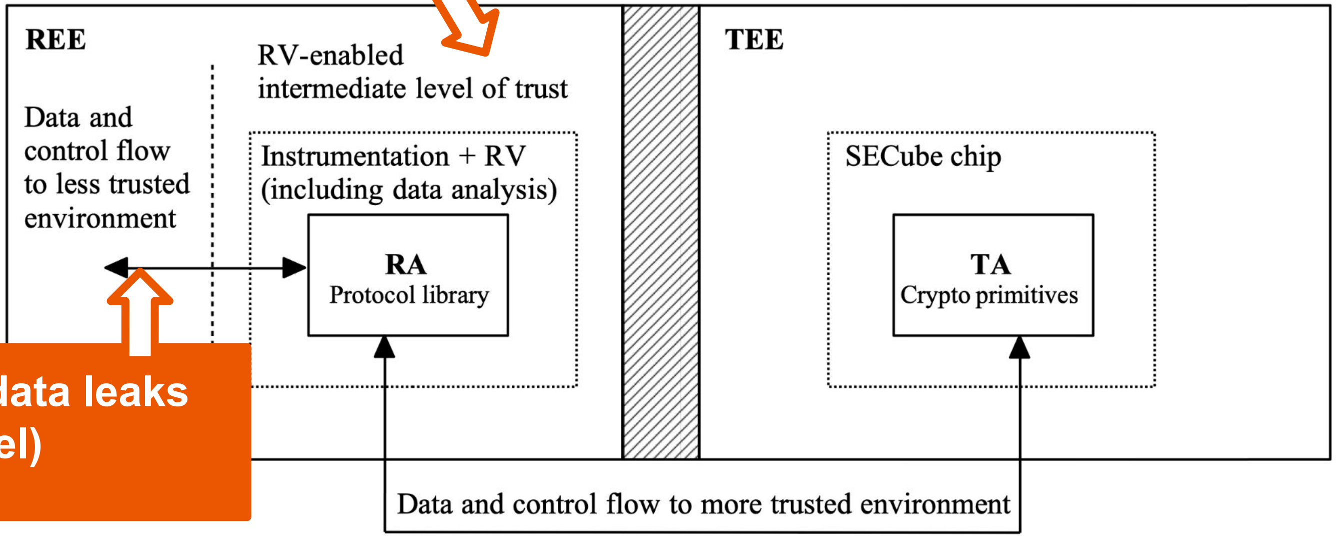


Monitor for data leaks (Medium level)

L-Università ta' Malta

Monitor code while executing (High level)

oring



Monitor for data leaks (Medium level)

L-Università ta' Malta

NATO OTAN

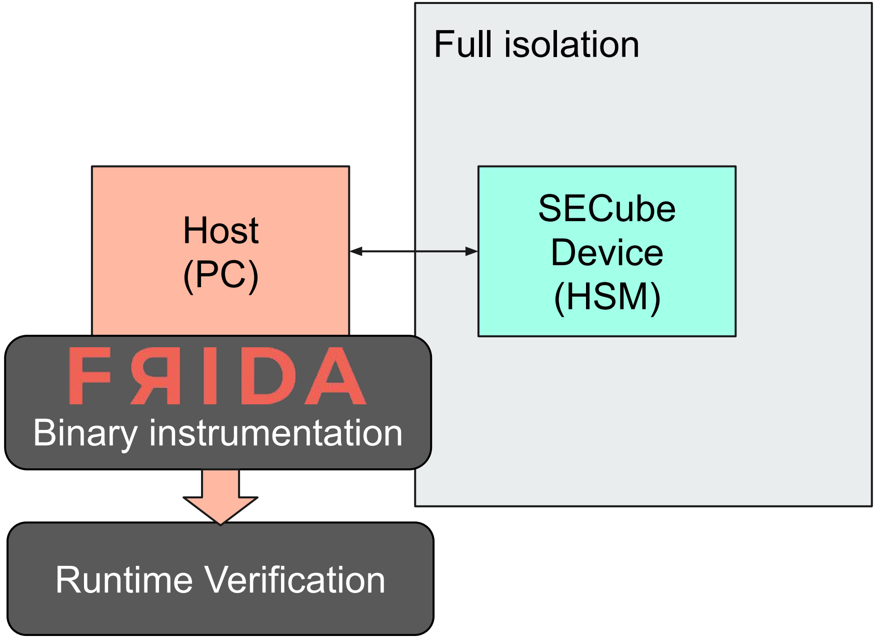
Summary

Control/Threat level	High	Medium	Low	H/W
RV-TEE	•Verify adherence to protocol design	•Isolated crypto execution •Exfiltration detection	•Software vulnerability detection *(offline)*	•Side-channel resistance •Tamper-evident
RV - function call tracing [4,68,69,81]	•Verify adherence to protocol design	–	–	–
RV - taint inference [67]	–	•Exfiltration detection	–	–
RV - information flow [3,70]	–	–	•Software vulnerability detection	–
HSM [10,21,30,47,55,72,73]	–	•Isolated crypto execution	–	•Side-channel resistance •Tamper-evident

How does it look in practice?

Hardware Security Module - SECube

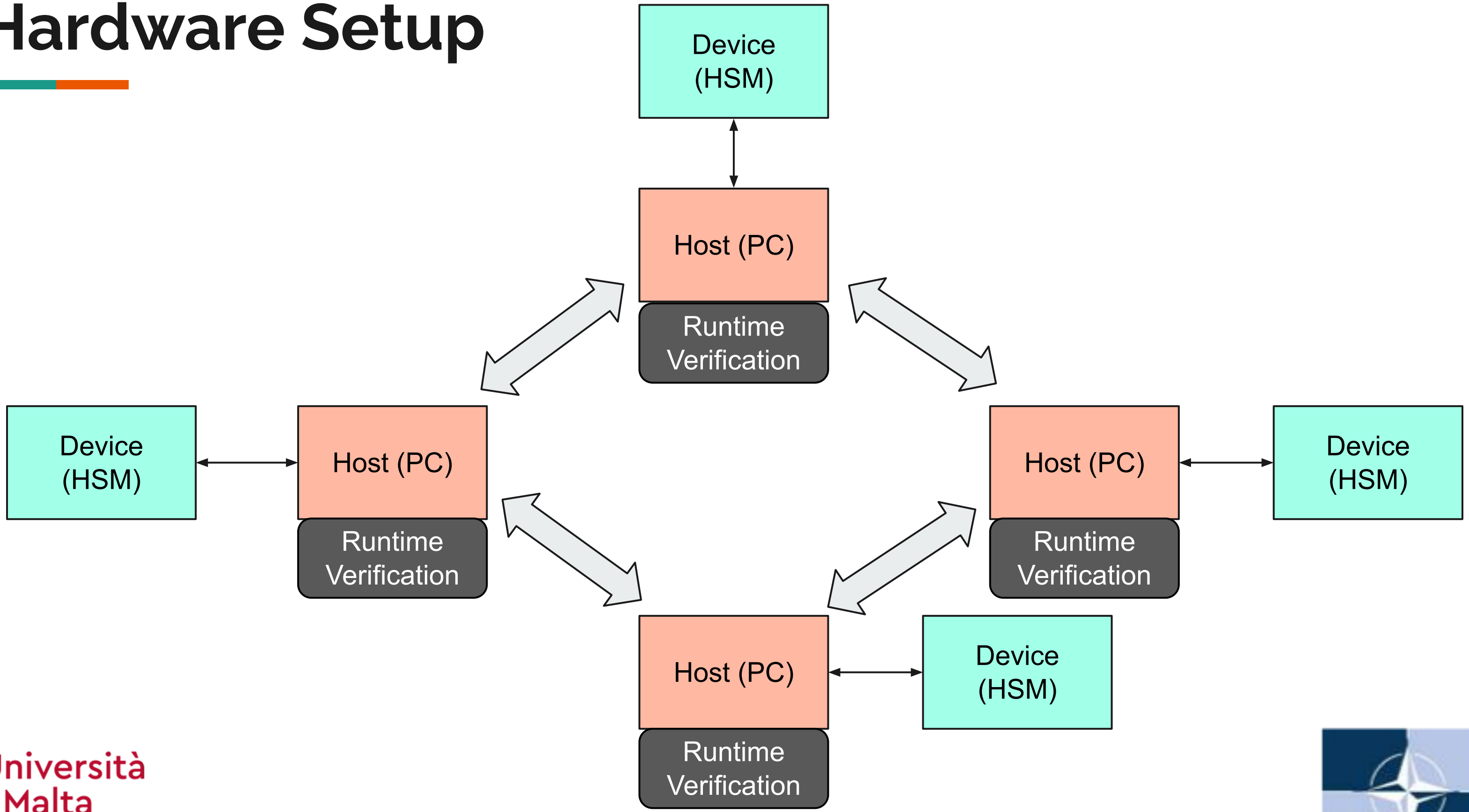
Hardware



L-Università ta' Malta

NATO OTAN

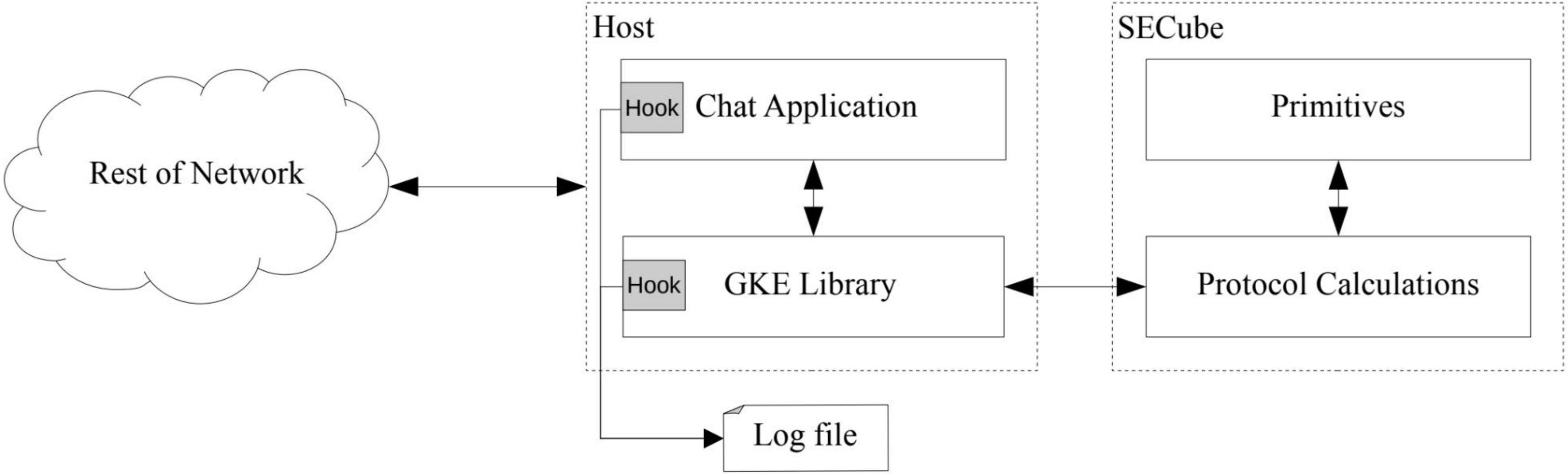
Hardware Setup



L-Università
ta' Malta

NATO
OTAN

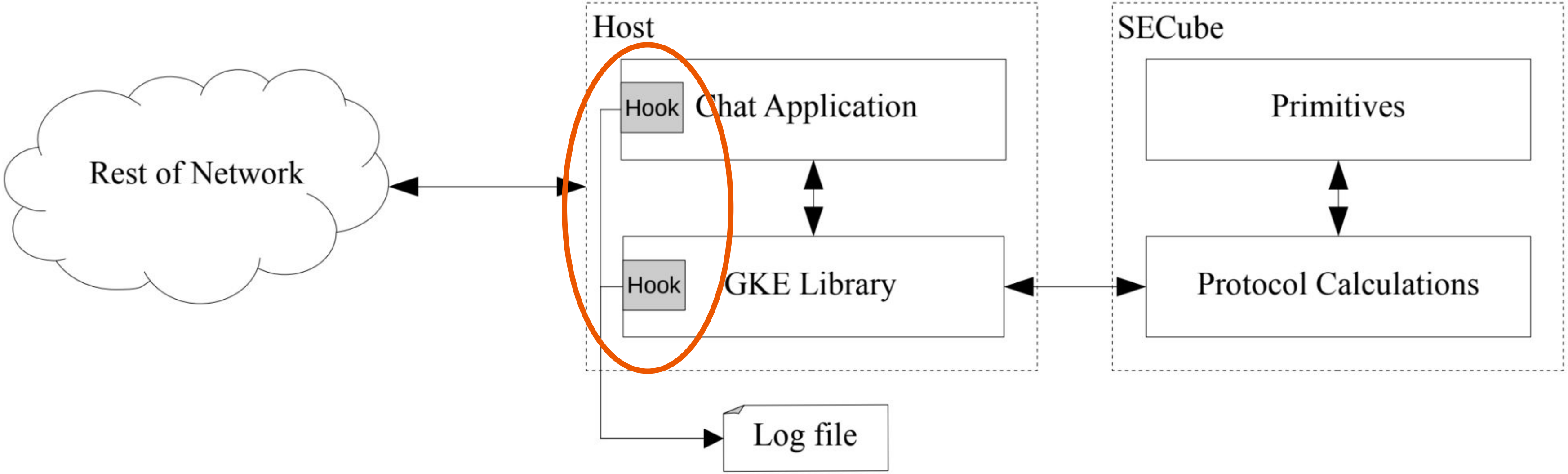
Looking Deeper at a Single Client



L-Università ta' Malta

NATO OTAN

Instrumentation



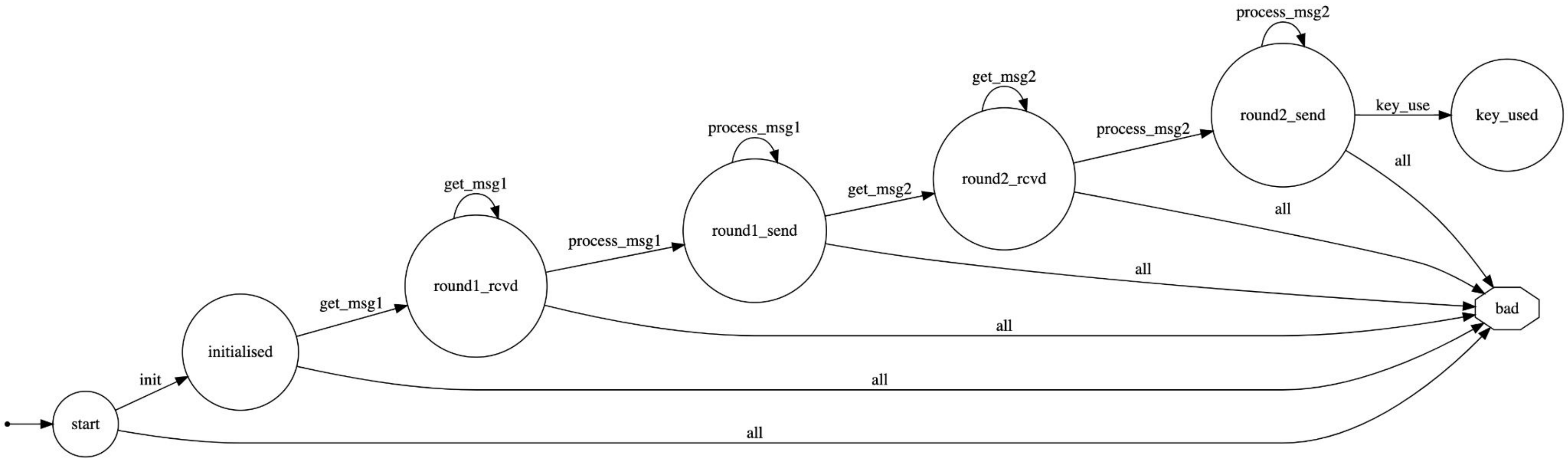
Log Output

```
[1935] 5642f6415b70  47 4b 45 7c 3e 20                                      GKE|>
[1935] 1625088325476 libpthread!read(fd=0x0, buf=0x7ffc1a22736f, count=0x1)
[1935] 1625088325476 libpthread!read() retVal: 0x1
[1935] 1625088325476 [HEXDUMP] out...
[1935] 7ffc1a22736f  2f                                                     /
[1935] 1625088325476 libpthread!read(fd=0x0, buf=0x7ffc1a2272df, count=0x1)
[1935] 1625088325476 libpthread!read() retVal: 0x1
[1935] 1625088325476 [HEXDUMP] out...
[1935] 7ffc1a2272df  72                                                     r
[1935] 1625088325476 libpthread!read(fd=0x0, buf=0x7ffc1a2272df, count=0x1)
[1935] 1625088325476 libpthread!read() retVal: 0x1
[1935] 1625088325476 [HEXDUMP] out...
[1935] 7ffc1a2272df  6f                                                     o
[1935] 1625088325476 libpthread!read(fd=0x0, buf=0x7ffc1a2272df, count=0x1)
[1935] 1625088325476 libpthread!read() retVal: 0x1
[1935] 1625088325476 [HEXDUMP] out...
[1935] 7ffc1a2272df  6f                                                     o
[1935] 1625088325476 libpthread!read(fd=0x0, buf=0x7ffc1a2272df, count=0x1)
[1935] 1625088325476 libpthread!read() retVal: 0x1
[1935] 1625088325476 [HEXDUMP] out...
[1935] 7ffc1a2272df  6d                                                     m
[1935] 1625088325476 libpthread!read(fd=0x0, buf=0x7ffc1a2272df, count=0x1)
[1935] 1625088325476 libpthread!read() retVal: 0x1
[1935] 1625088325476 [HEXDUMP] out...
[1935] 7ffc1a2272df  20
[1935] 1625088325476 libpthread!read(fd=0x0, buf=0x7ffc1a2272df, count=0x1)
[1935] 1625088325476 libpthread!read() retVal: 0x1
```

High level monitors

Property layers	Chat app	GKE Library	All (incl. Primitives)
Assertion	Printable decrypted characters	Sensitive data scrubbed	Valid function parameters and returns
Temporal	Chatroom lifecycle, standard sockets		Correct function call sequence
Hyper		Randomness quality	

Formal Specification

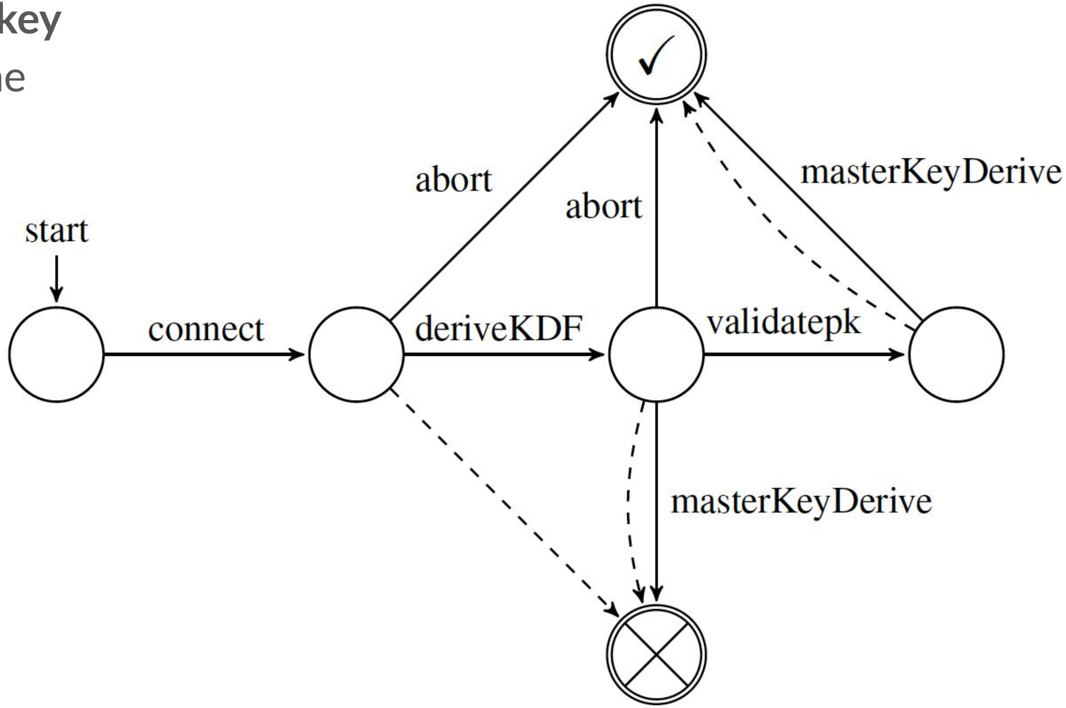


L-Università ta' Malta

NATO OTAN

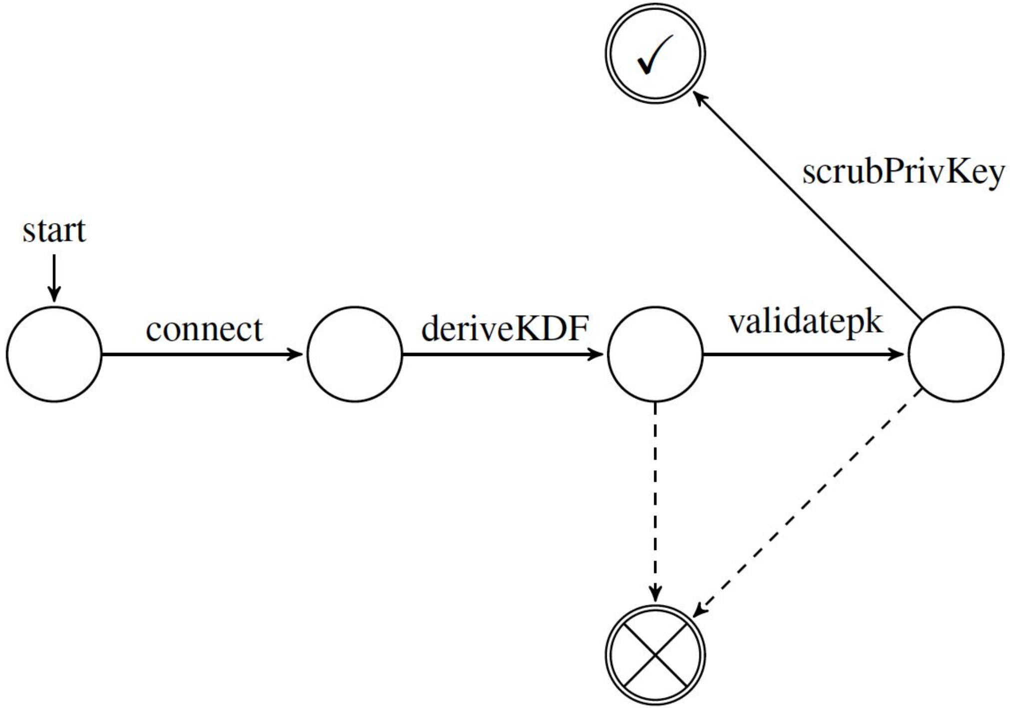
Properties verified (High level) on ECDHE

Validation of remote peer's **public key** on each exchange is done (unless the session is aborted)



Properties verified (High level) on ECDHE

Once master secret is established, private keys should be **scrubbed from memory**



Monitoring Overheads

– Scenario A: 3 clients involved, with client `id=1` creating a room (following the protocol steps for an initiator participant U_0).
– Scenario B: 3 clients involved, with client `id=1` joining the room (following the protocol steps for a non-initiator participant $U_{1 \leq i \leq n}$).

The scenarios include 20 and 13 seconds of thread sleeps respectively to mimic a realistic chat. This will be factored in in the results discussion.

Monitoring Overheads

A - Creating a chat room
B - Joining a chat room

Time (s)	Without SEcube™			Using SEcube™		
Scenario	A	B	All	A	B	All
Non-instrumented	20.02	13.01	33.03	20.18	13.27	33.45
Instrumented	20.44	14.39	34.83	21.30	13.68	34.98

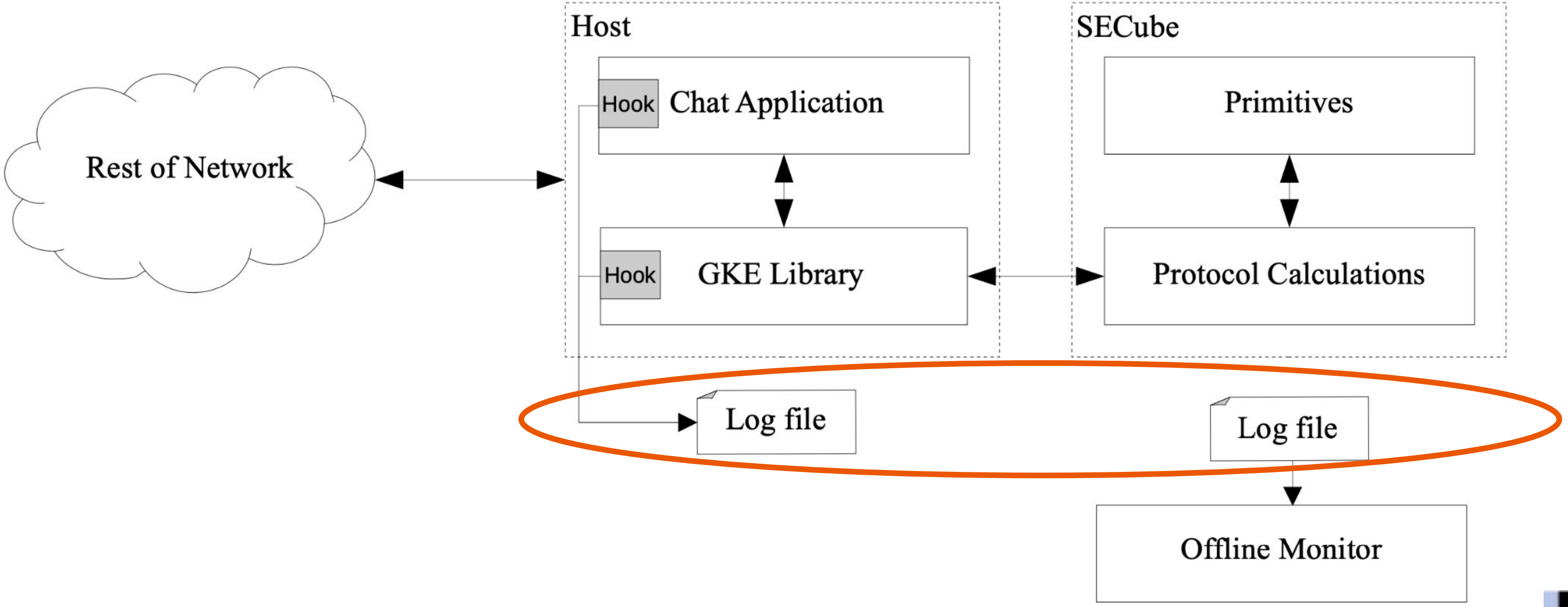
Monitoring Overheads

A - Creating a chat room
B - Joining a chat room

Time (s)	Without SEcube™			Using SEcube™			
Scenario	A	B	All	A	B	All	
Non-instrumented	20.02	13.01	33.03	20.18	13.27	33.45	1%
Instrumented	20.44	14.39	34.83	21.30	13.68	34.98	0.4%
Increase	2%	11%	5%	6%	3%	5%	

Instrumentation is more expensive than HSM

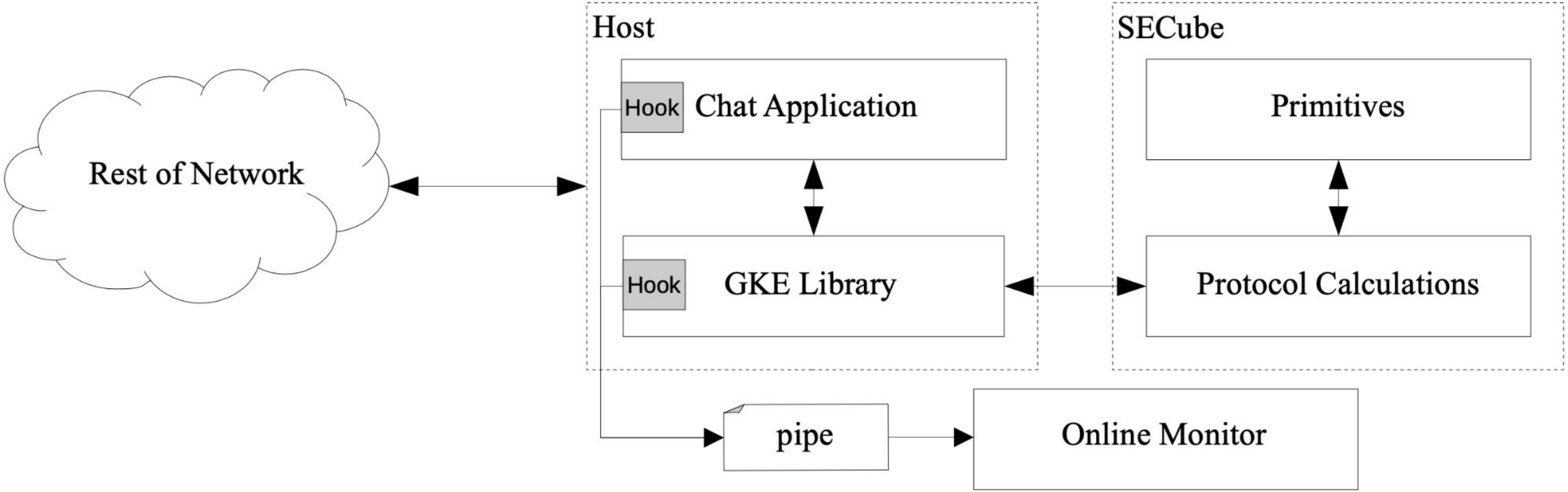
Running the Monitor Offline



L-Università ta' Malta

NATO OTAN

Switching to Online Monitoring



L-Università ta' Malta

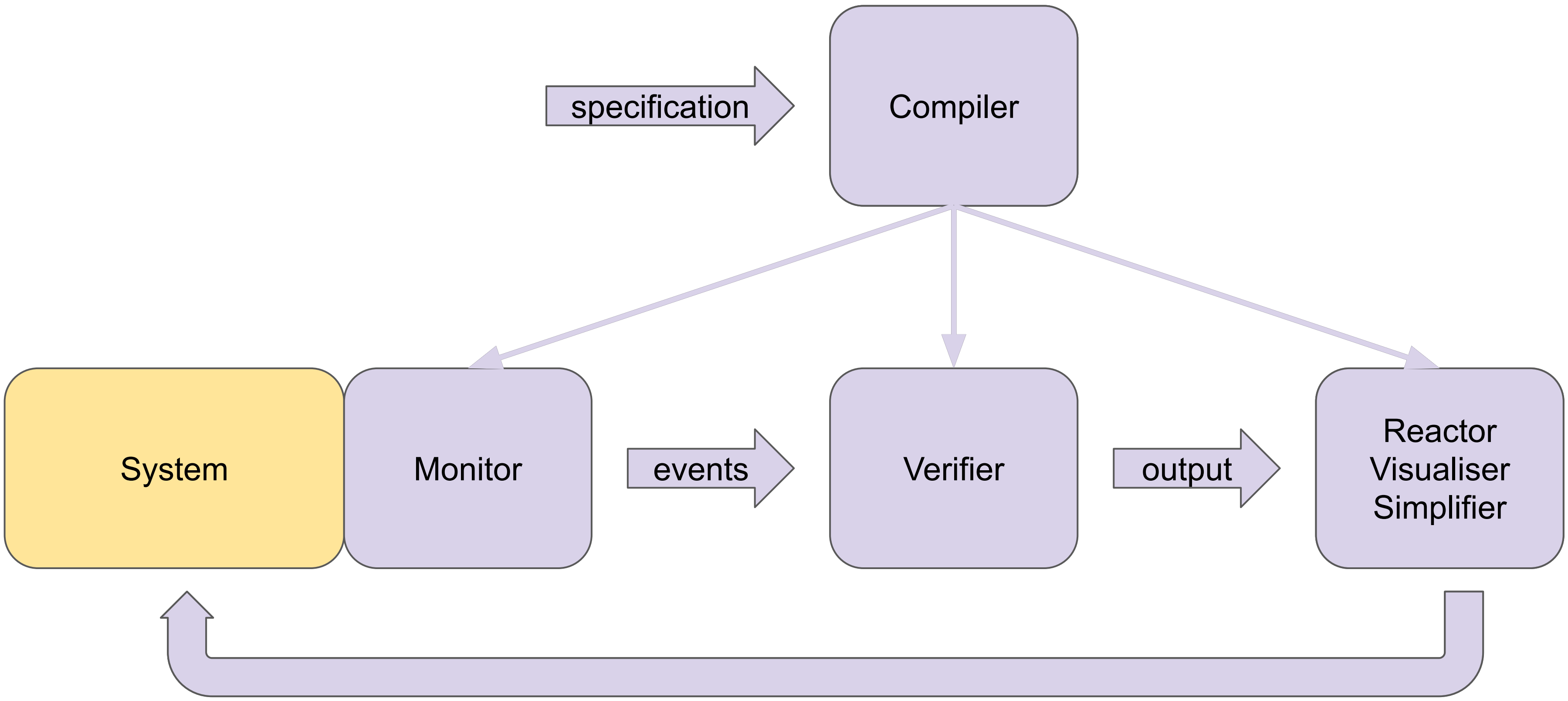
NATO OTAN

```
natosps@natosps-Z87-HD3:~/git/GKEdemo/GKE/Instrumentation/Injection$ python3 injectRV.py ../../bin/chat --id 3 --repeater 147.175.106.130
FIFO named '/tmp/pipe_241798_to_larva' is created successfully.
*******************************************************************
* You are now connected to the server.                            *
* You can now create a chat room or enter an existing chat room.  *
* For available commands, please type /help.                      *
*******************************************************************
GKE|> waiting for chat app
/room new U 4
Creating room 'U'
-----------------------------------------------------------------
CONGRATULATIONS! The room 'U' was created for users {3,4}.
A shared secret key for all room users was established by the Quantum-Future Group Authenticated Key Exchange Protocol.
From now on, the shared secret key will be used with AES CCM 128 to encrypt communication between room users.
You can now enter the room and start sending encrypted messages.
-----------------------------------------------------------------
GKE|> RV:: *1* Initialised
RV:: *a* init_protocol_run_env called as expected
RV:: *b0* init_participant called as expected
RV:: *b1* init_participant called as expected
RV:: *c* round_one called as expected
RV:: *d* load_pw called as expected
RV:: *e* generate_beta called as expected
RV:: *f* calculate_g called as expected
RV:: *2* During the key exchange protocol (executed during room creation) the correct number of messages were received in round 1.
RV:: *g* round_two called as expected
RV:: *h* generate_k called as expected
RV:: *i* extract_result called as expected
RV:: *j1* kem_enc called as expected
RV:: *k1* calculate_shared_value called as expected
RV:: *l1* generate_MAC_init called as expected
RV:: *3* During the key exchange protocol (executed during room creation) the correct number of messages were sent in round 1.
RV:: *4* During the key exchange protocol (executed during room creation) the correct number of messages were received in round 2.
RV:: *5* During the key exchange protocol (executed during room creation) the correct number of messages were sent in round 2.
RV:: *m* round_two_finalize called as expected
RV:: *n* generate_MAC_non_init called as expected
/room enter U
-----------------------------------------------------------------
You have entered the room 'U'
You can now type a message directly to the command line.
The message will be encrypted and will be sent to all room users.
For other available actions, please type /roomhelp
-----------------------------------------------------------------
GKE|U> test
[U]3: test
GKE|U> RV:: *6* The previously received message was decrypted using the shared secret key established during the creation of the room.
```

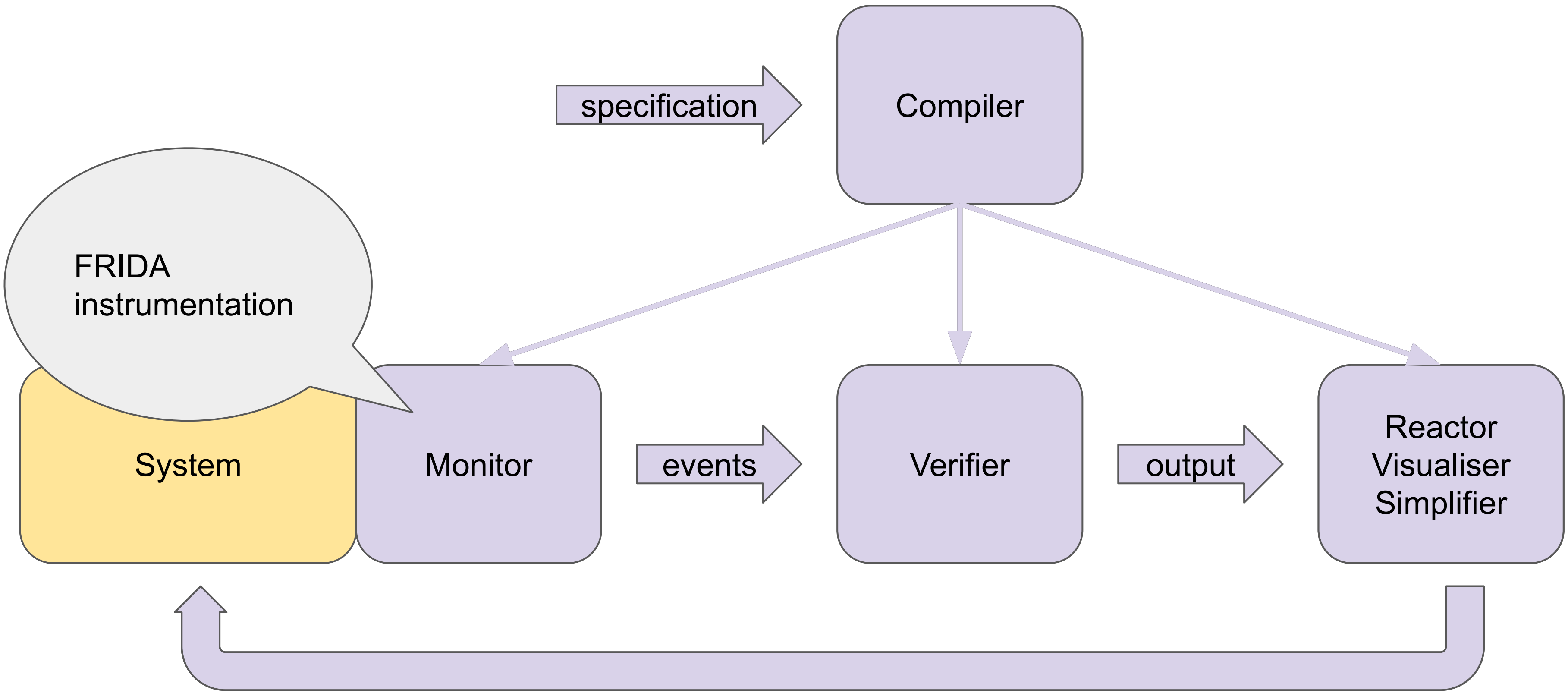
Monitor Alert

```
GKE|room1> hello
[room1]3: hello
GKE|room1> RV:: *6* The previously received message was decrypted using the shar
ed secret key established during the creation of the room.
máme nový protokol!
GKE|room1> RV::*!WRONG!* Found non-ASCII characters: [109, -61, -95, 109, 101, 3
2, 110, 111, 118, -61, -67, 32, 112, 114, 111, 116, 111, 107, 111, 108, 33]
[room1]3: máme nový protokol!
```

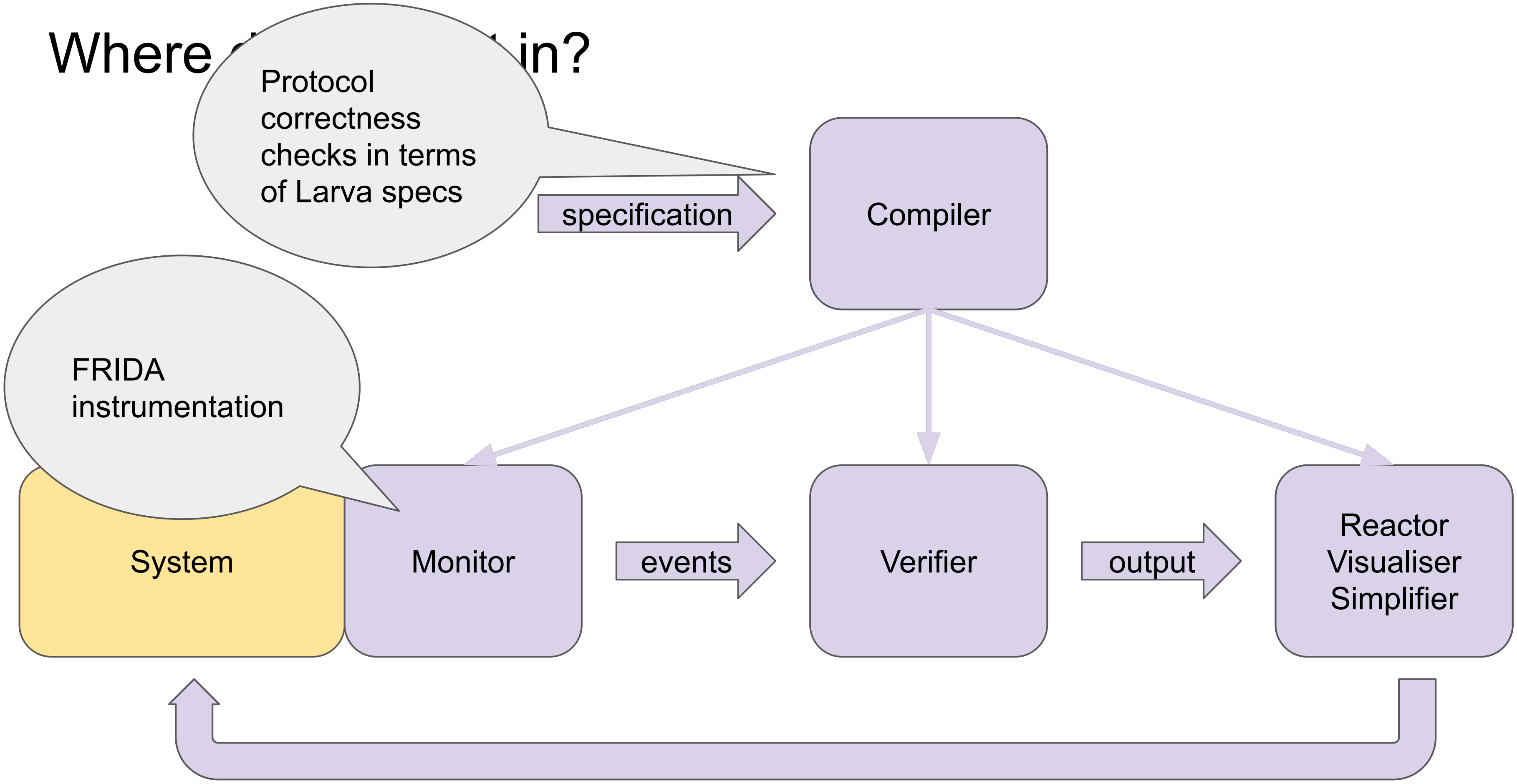
Where does RV fit in?



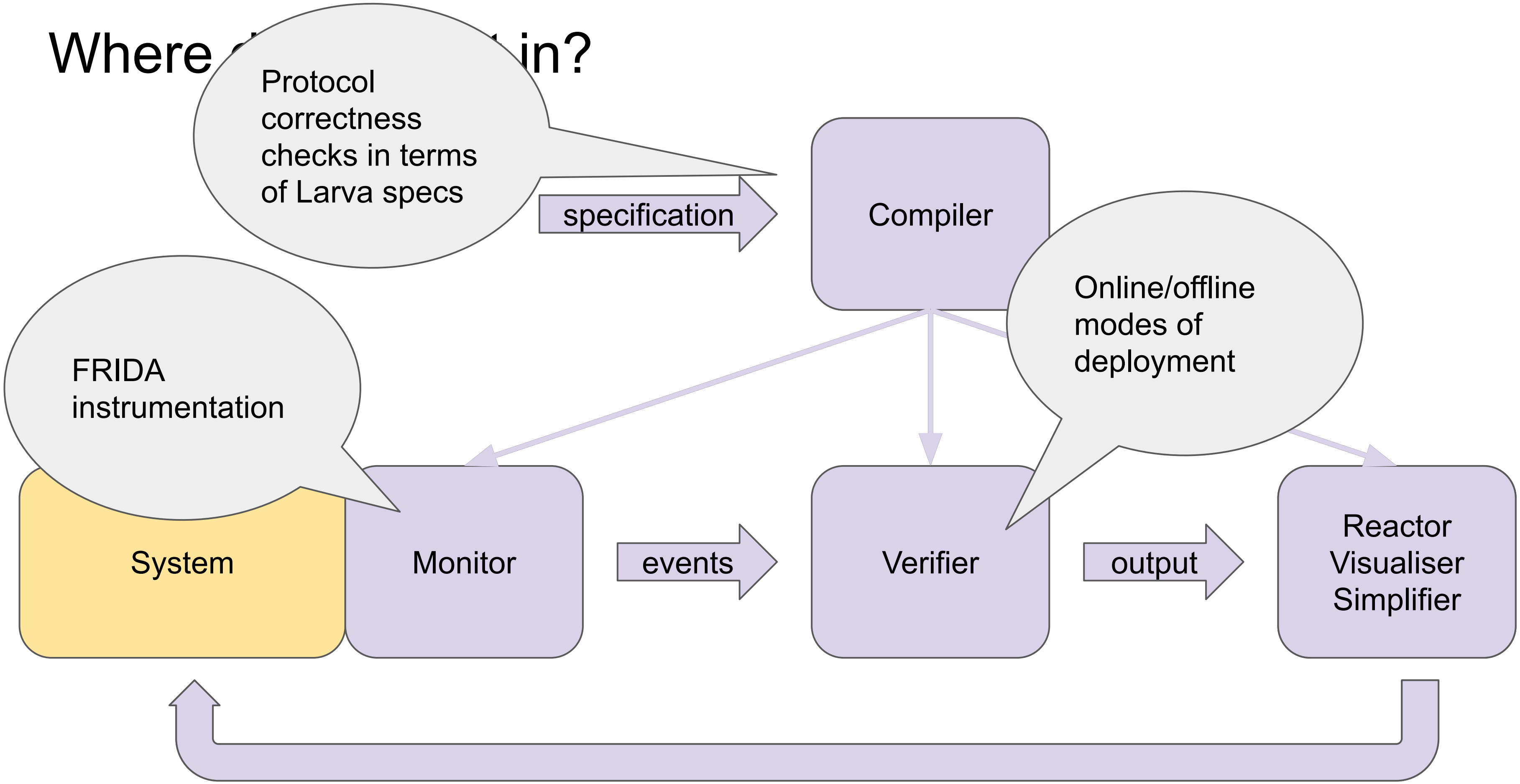
Where does RV fit in?



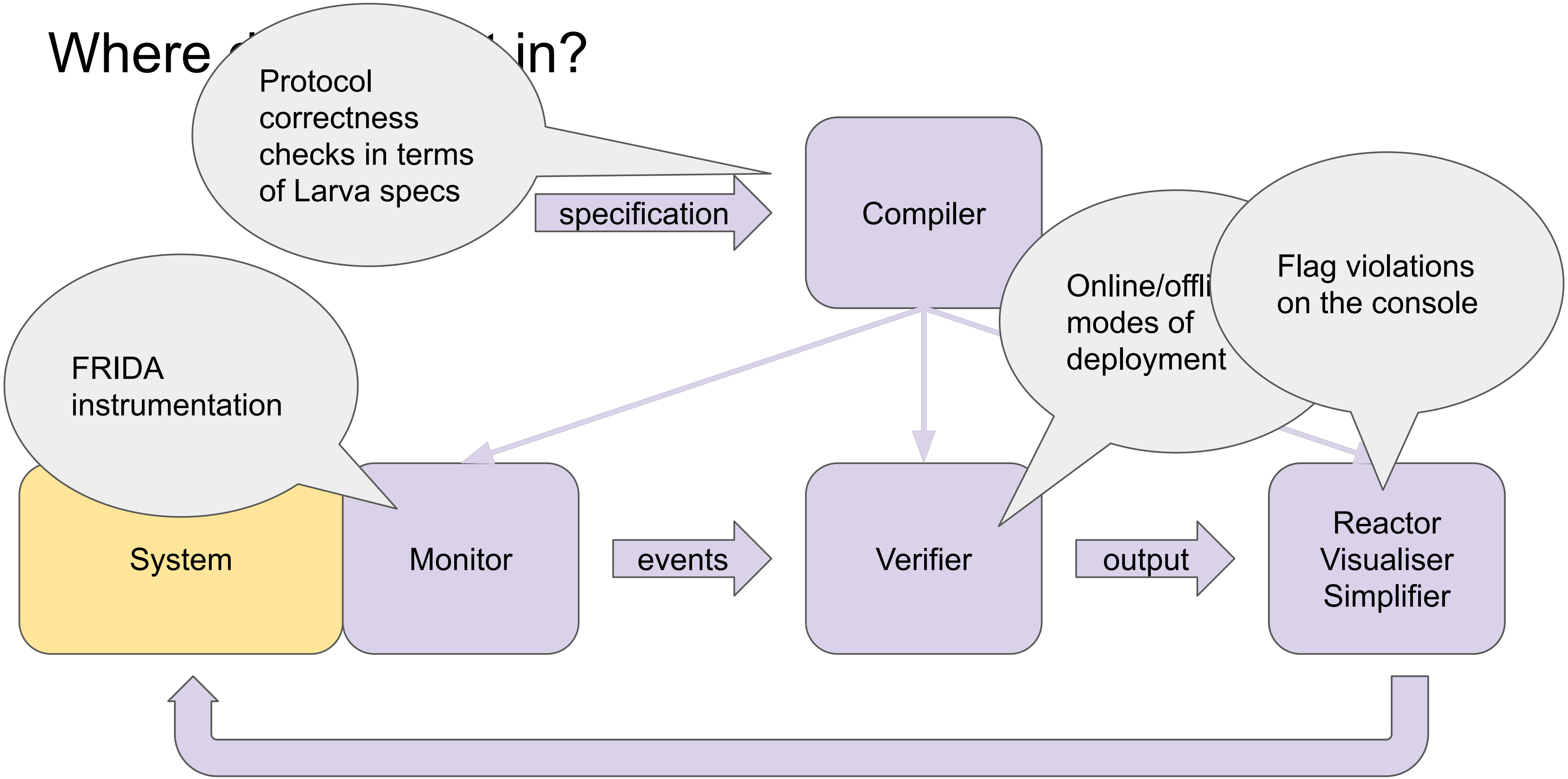
Where … in?


Protocol correctness checks in terms of Larva specs
specification
Compiler
FRIDA instrumentation
System
Monitor
events
Verifier
output
Reactor
Visualiser
Simplifier

Where … in?



Protocol correctness checks in terms of Larva specs
specification
Compiler
Online/offline modes of deployment
FRIDA instrumentation
System
Monitor
events
Verifier
output
Reactor
Visualiser
Simplifier


Where
in?
Protocol correctness checks in terms of Larva specs
specification
Compiler
Online/offli
modes of deployment
Flag violations on the console
FRIDA instrumentation
System
Monitor
events
Verifier
output
Reactor
Visualiser
Simplifier

Future/Ongoing Work

#	Context	Tech	Instrumentation	Data aspect	On/offline
1	Firefox	C++	DBI (Frida)	Taint inference on outgoing data	off
2	Paramiko	Python	AOP (aspectlib)	Limited to parameter checking	off
3	Chat app	C++	DBI (Frida)	Monitoring incoming data	on (async)

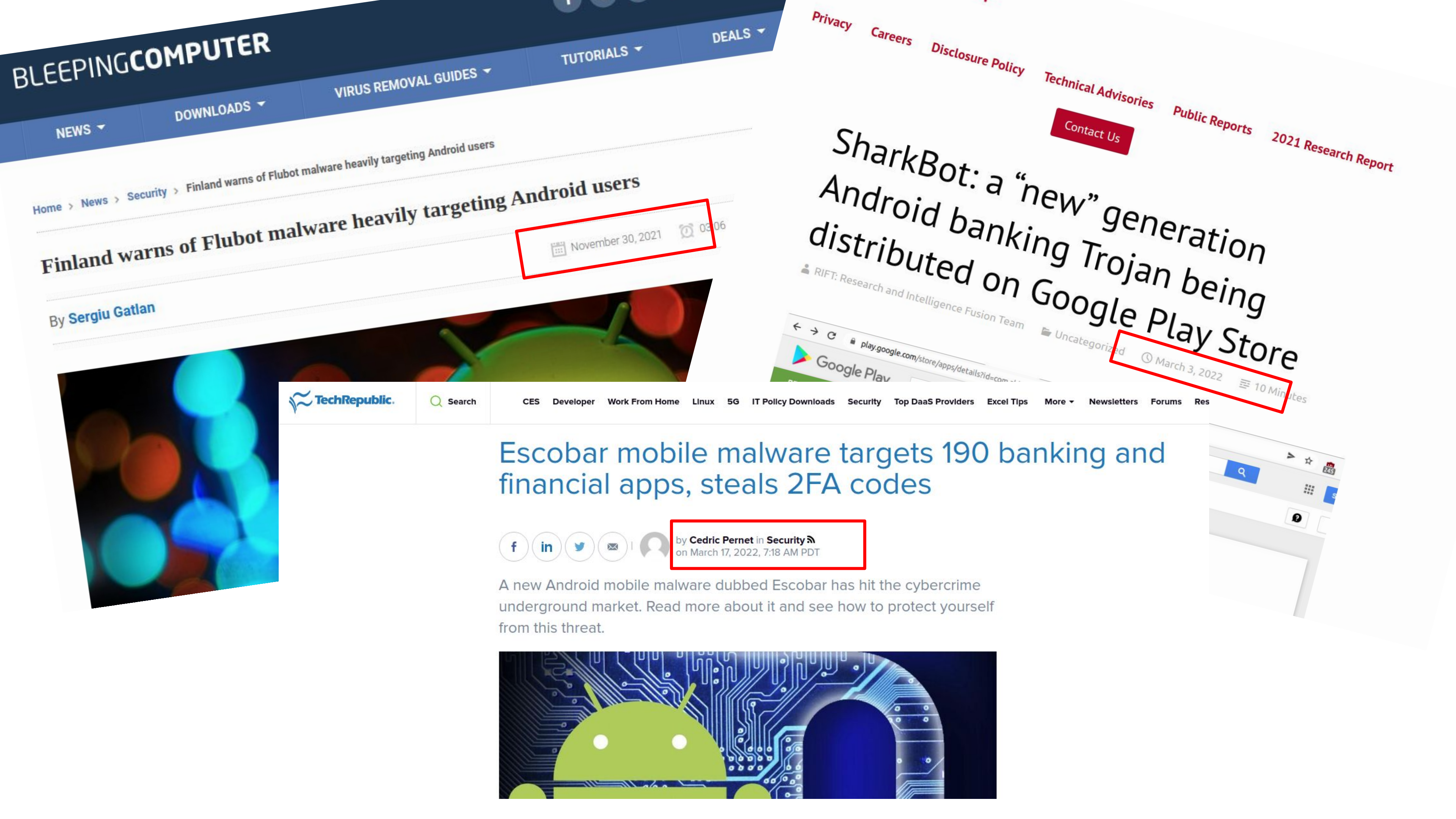
Future/Ongoing Work

Protecting the monitor

- The monitor is executed in a protected environment (RunC Container)
- The logs are encrypted and stored in temper-evident file system (SEALFS)

Cyber Security (2)
Extracting Evidence

(part of Horizon 2020
LOCARD project)


BLEEPINGCOMPUTER
NEWS
DOWNLOADS
VIRUS REMOVAL GUIDES
TUTORIALS
DEALS
Home > News > Security > Finland warns of Flubot malware heavily targeting Android users
Finland warns of Flubot malware heavily targeting Android users
November 30, 2021
By Sergiu Gatlan
Privacy
Careers
Disclosure Policy
Technical Advisories
Public Reports
2021 Research Report
Contact Us
SharkBot: a "new" generation Android banking Trojan being distributed on Google Play Store
RIFT: Research and Intelligence Fusion Team
Uncategorized
March 3, 2022
TechRepublic.
Search
CES
Developer
Work From Home
Linux
5G
IT Policy Downloads
Security
Top DaaS Providers
Excel Tips
More
Newsletters
Forums
Escobar mobile malware targets 190 banking and financial apps, steals 2FA codes
by Cedric Pernet in Security
on March 17, 2022, 7:18 AM PDT
A new Android mobile malware dubbed Escobar has hit the cybercrime underground market. Read more about it and see how to protect yourself from this threat.

Stealthy Malware - Living Off the Land (LOtL)

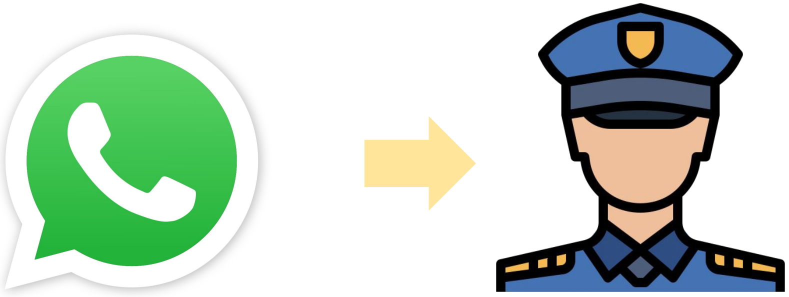
- Delegate sensitive tasks (e.g. sending messages) to benign apps
- Leave little to no evidence behind (no suspicious permissions needed)
- Cannot avoid executing in memory

Assumptions

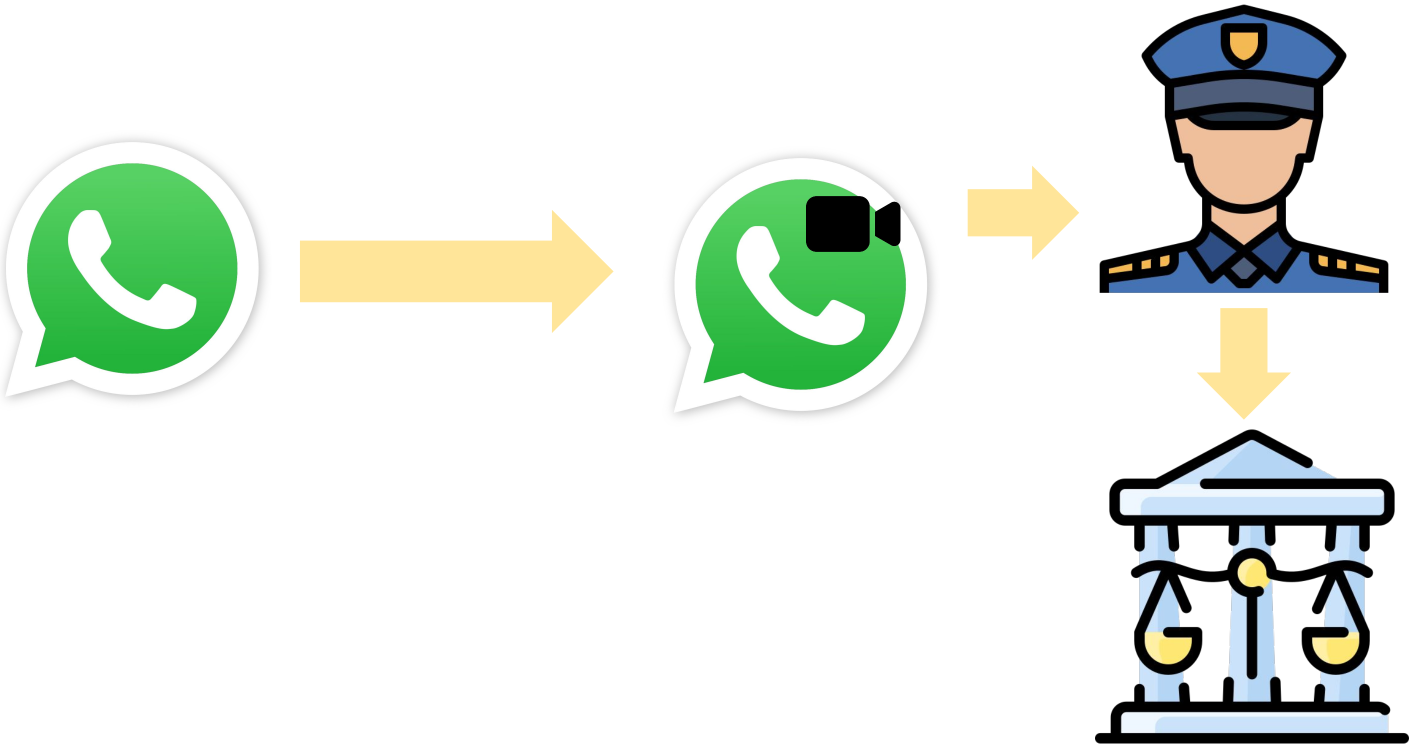
- We don’t modify Android
- We don’t modify the app
- We want an approach which is easy to use across apps and app versions

Whatsapp Example

Could Whatsapp be sending messaging without me knowing it?



Instrumenting Whatsapp



MobFor (ii)

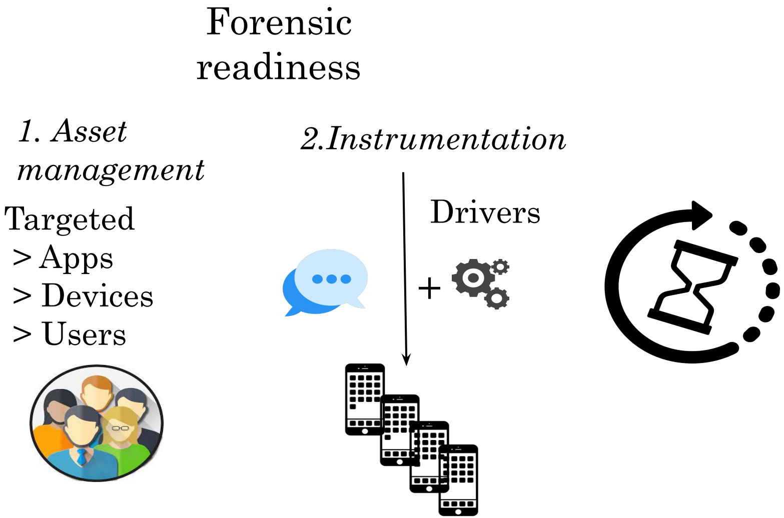
Forensic readiness

1. Asset management

Targeted
> Apps
> Devices
> Users

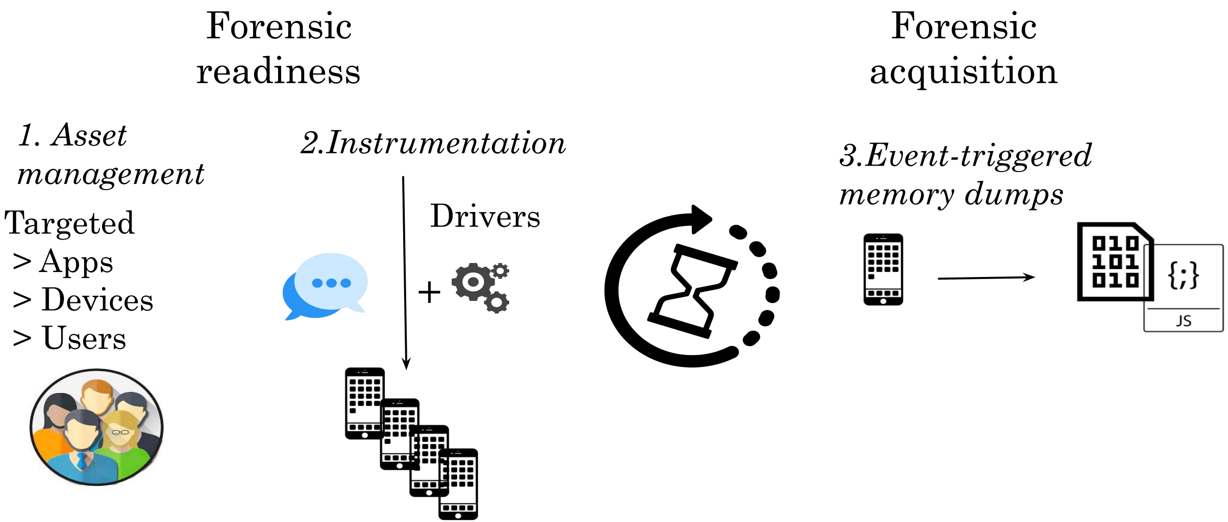
This project has received financial support from the European Union Horizon 2020 Programme under grant agreement no. 832735.

MobFor (ii)



This project has received financial support from the European Union Horizon 2020 Programme under grant agreement no. 832735.

MobFor (ii)



This project has received financial support from the European Union Horizon 2020 Programme under grant agreement no. 832735.

MobFor (ii)

Forensic readiness

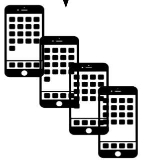
1. Asset management

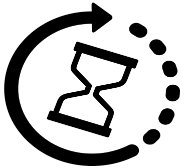
Targeted
> Apps
> Devices
> Users

2.Instrumentation

Drivers

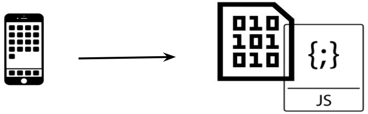
+





Forensic acquisition

3.Event-triggered memory dumps

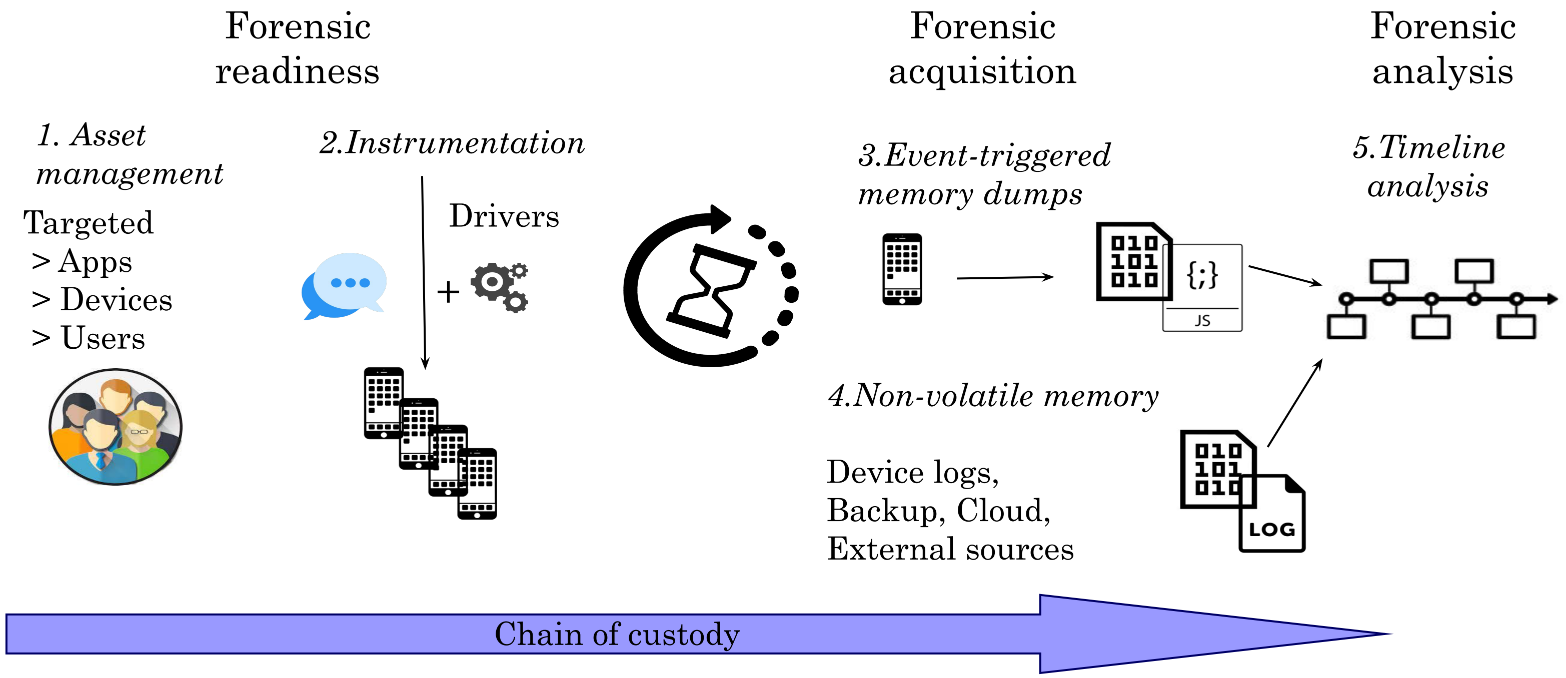


4.Non-volatile memory

Device logs,
Backup, Cloud,
External sources

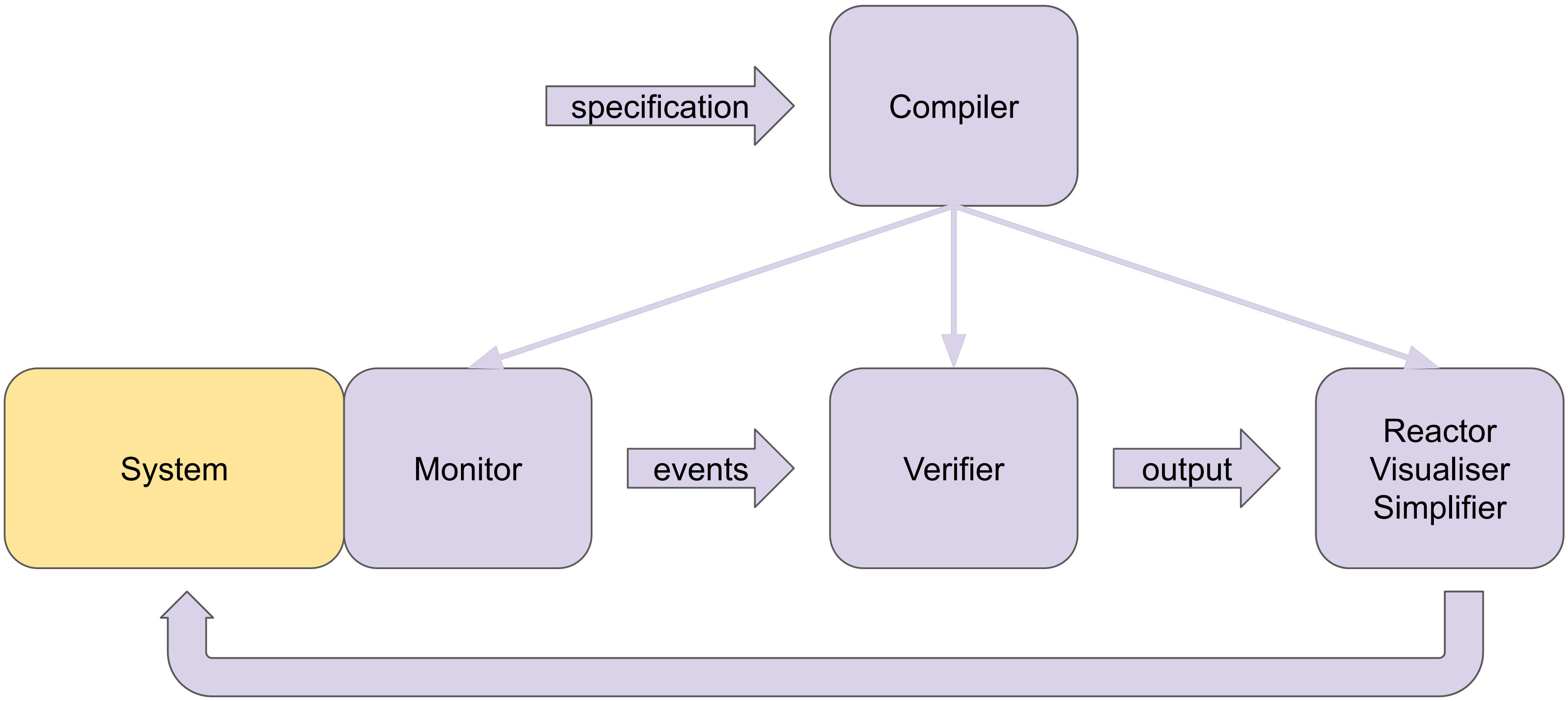
This project has received financial support from the European Union Horizon 2020 Programme under grant agreement no. 832735.

MobFor (ii)

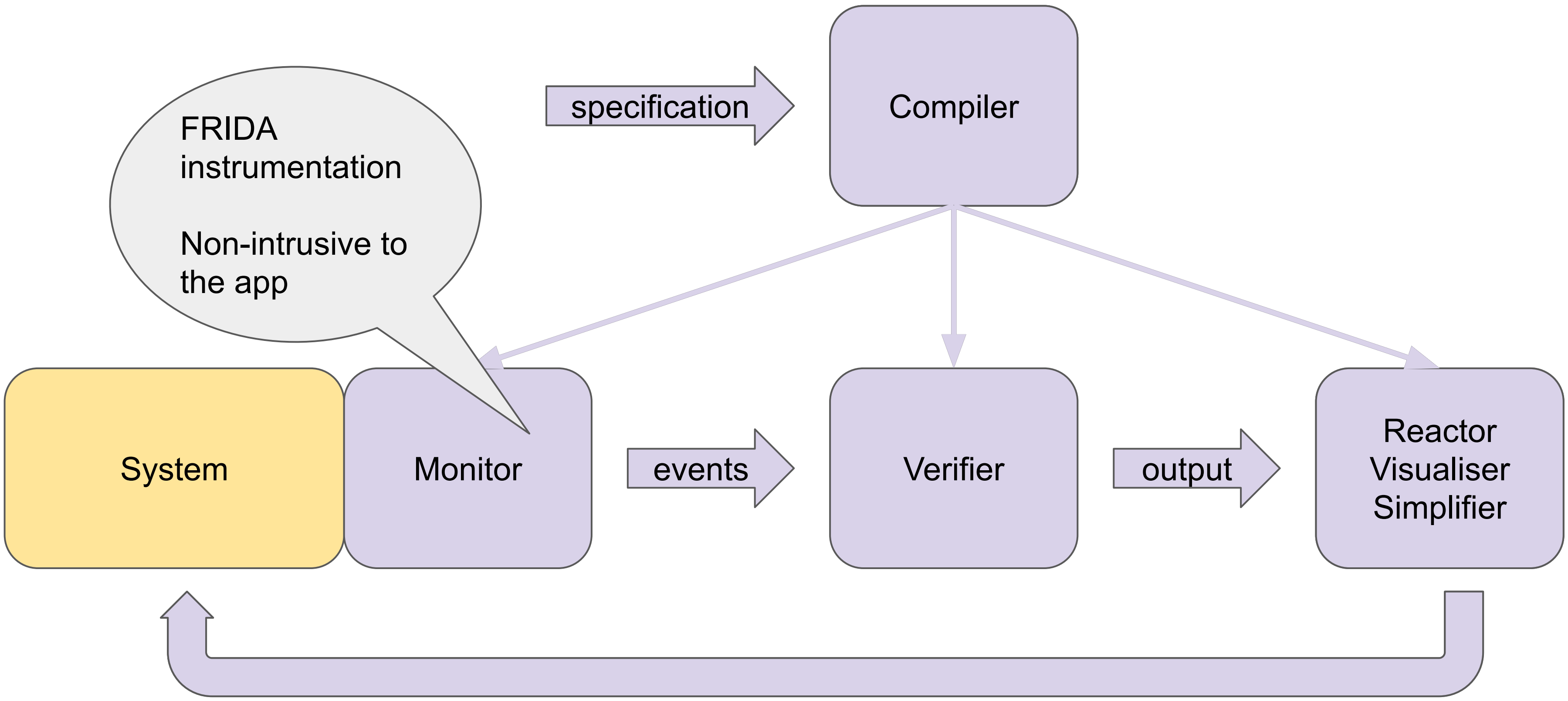


This project has received financial support from the European Union Horizon 2020 Programme under grant agreement no. 832735.

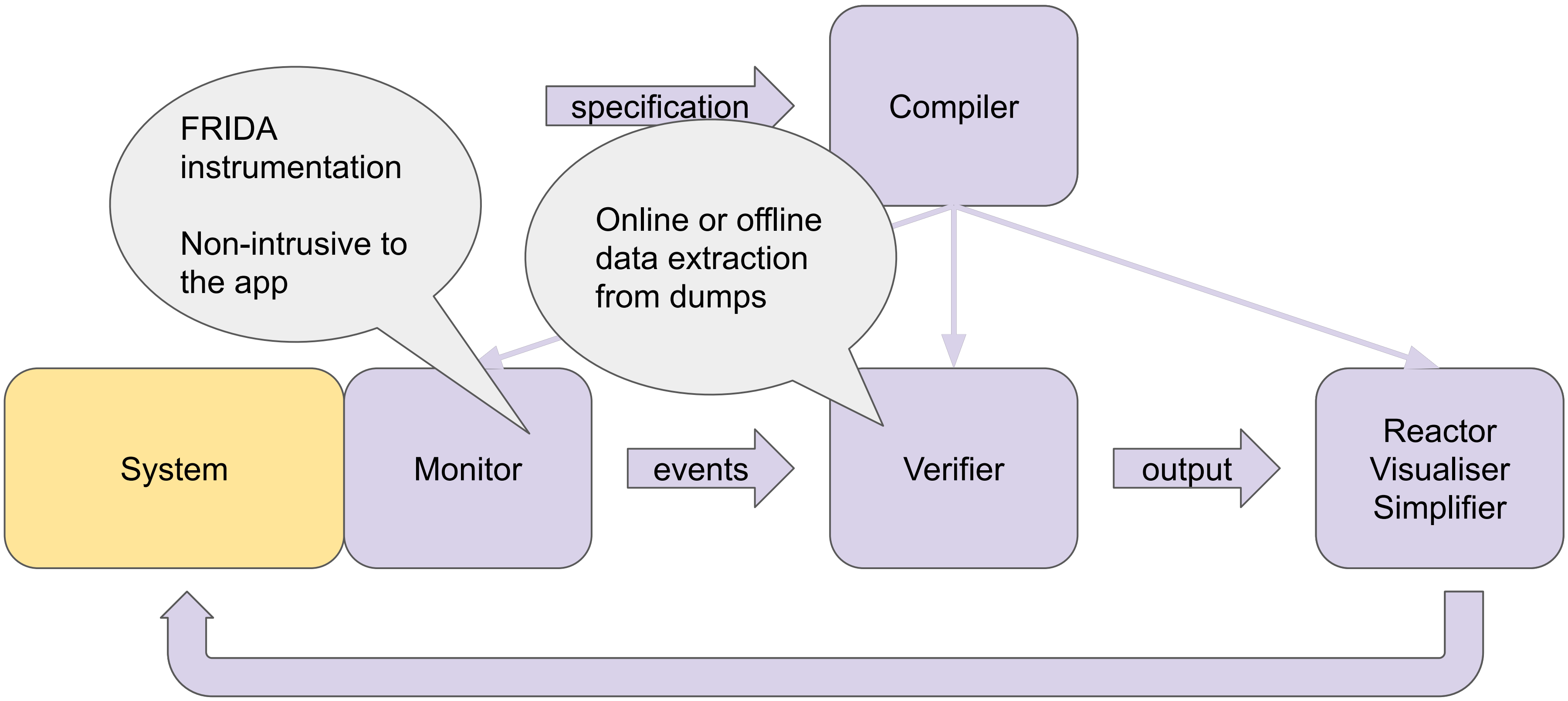
Where does RV fit in?



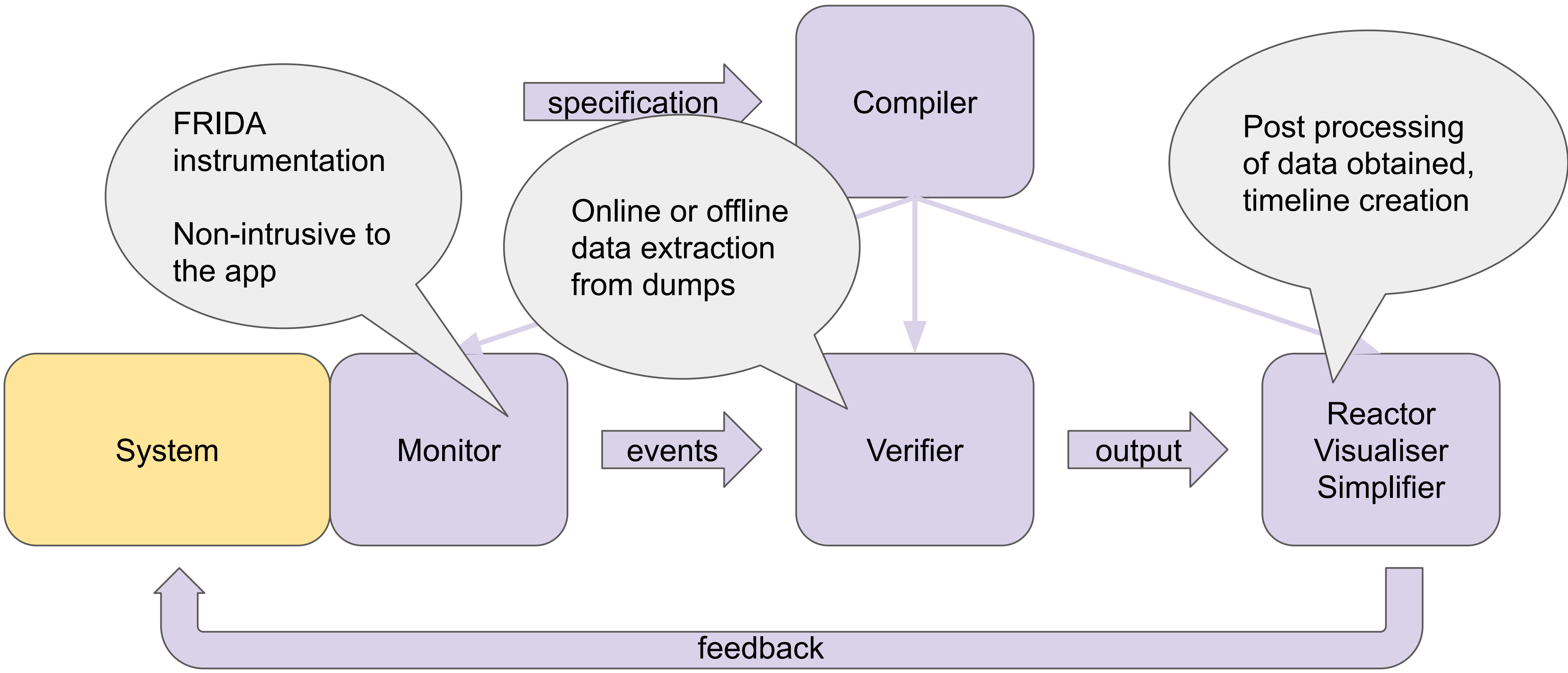
Where does RV fit in?



Where does RV fit in?



Where does RV fit in?



Future/Ongoing Work

Making agents more generic (using infrastructure-based trigger points)

Hosting app in virtual app instead of repackaging

Trying the same approach on financial apps

Adding anomaly detection to show value added of newly logged events

Conclusions

Conclusions

RV in itself offers two main ideas:

- Formal specifications
- Separation of concerns

The nice thing about RV is that it has a lot to offer to different areas

Two main projects:

- Securing a cryptographic protocol (in conjunction with hardware)
- Extracting events from memory