universität innsbruck

UNIVERSITÄT LIECHTENSTEIN

universität ulm

Model-based Generation of Attack-Fault Trees

SAFECOMP 2023

R. Groner, T. Witte, A. Raschke, S. Hirn, **I. Pekaric**, M. Frick, M. Tichy and M. Felderer

Combined Safety and Security Analysis

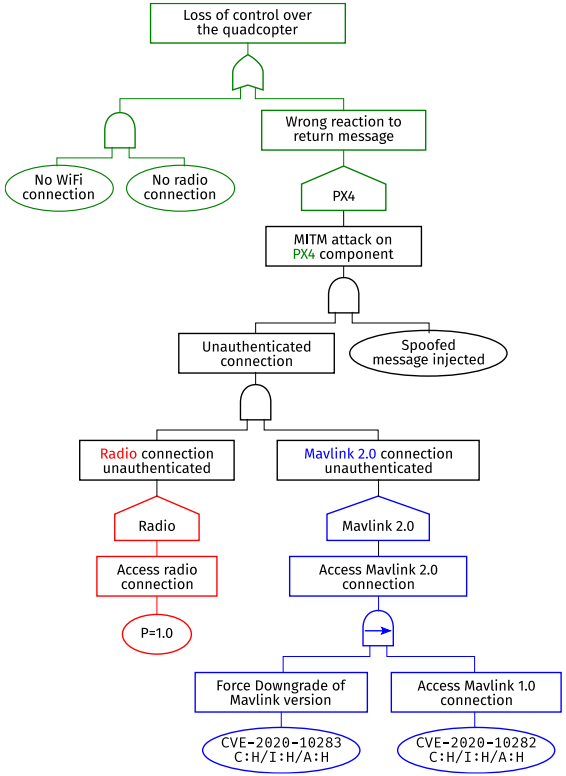
- Safety and Security considered separate concerns in the past
- Increasing software and interface complexity in SAS and CPS
  - The adaptation itself can be a target
- Safety and Security are now strongly interrelated:
  - Security flaws can cause safety hazards.
  - Safety mechanisms may affect security.
  - Adaptation changes safety and security properties.

https://www.bleepingcomputer.com/news/security/medtronic-urgently-recalls-insulin-pump-controllers-over-hacking-concerns/

https://abcnews.go.com/Technology/cyberattacks-reported-us-airports/story?id=91287965

Attack-Fault-Trees / State of the Art

- Combination of attack and fault trees
- Can be extended to include time and other metrics
- Existing analysis approach/translation to PTFAs [1][2]



[1] André, É., Lime, D., Ramparison, M., & Stoelinga, M. (2021). Parametric analyses of attack-fault trees. *Fundamenta Informaticae*, *182*(1), 69-94.

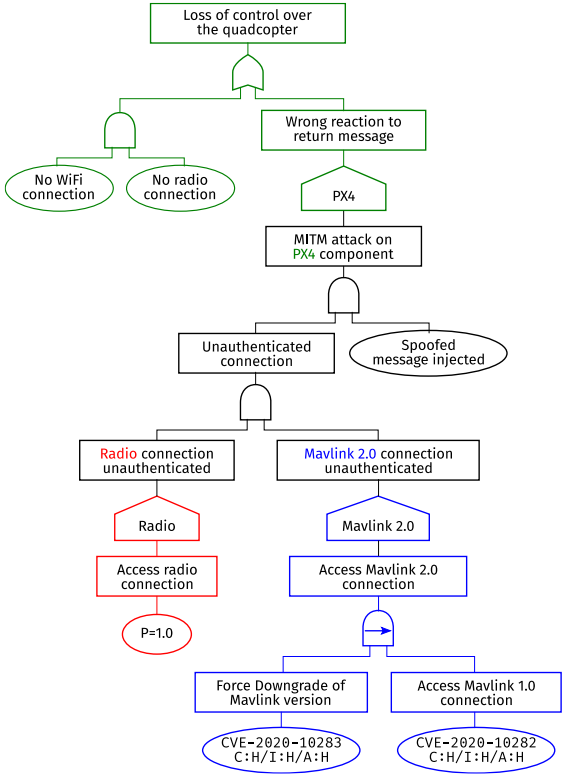
[2] Kumar, R., & Stoelinga, M. (2017, January). Quantitative security and safety analysis with attack-fault trees. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)* (pp. 25-32).

Attack-Fault-Trees and SAS

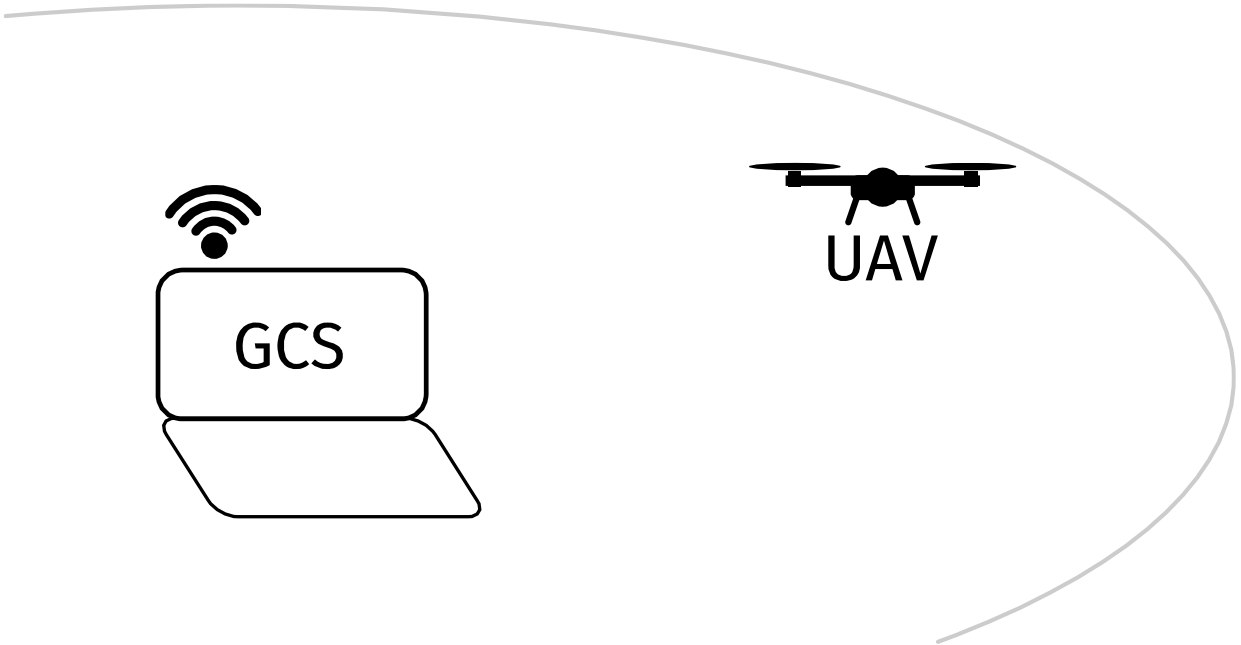
- AFT modeled by hand, unclear how changes to the system change the model
- Often very abstract, unclear how basic events relate to e.g. system vulnerabilities

Models should:

- closely relate to the system (and change with it)
- bridge the abstraction gap
- automatically translate to AFTs

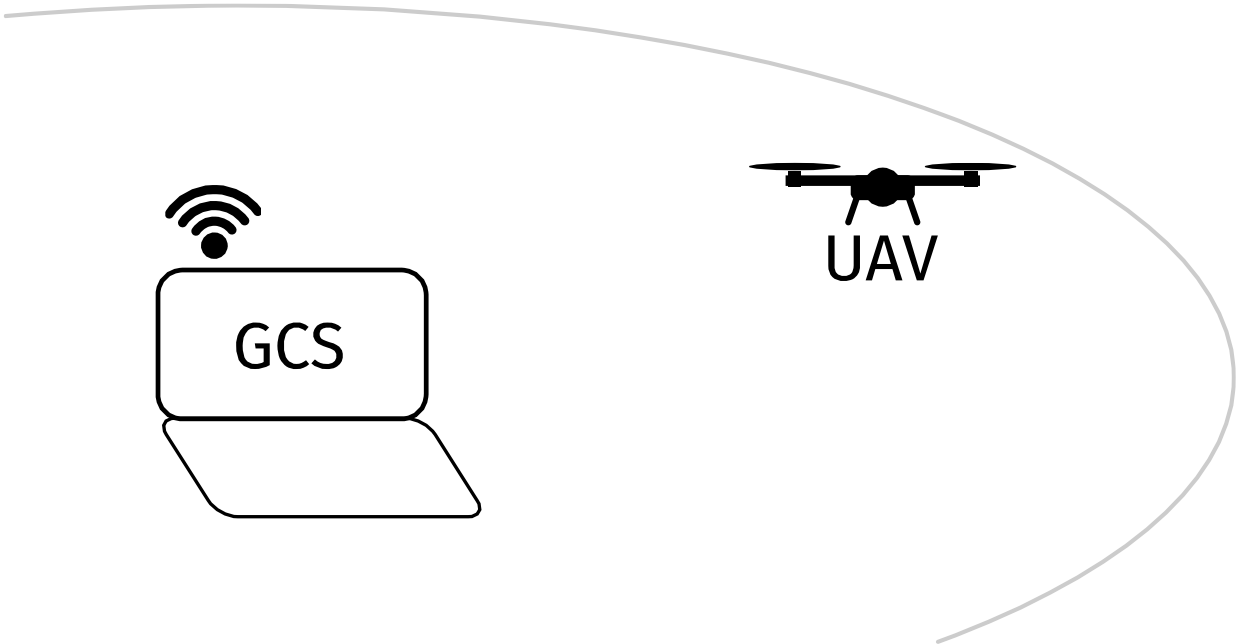


Running Example



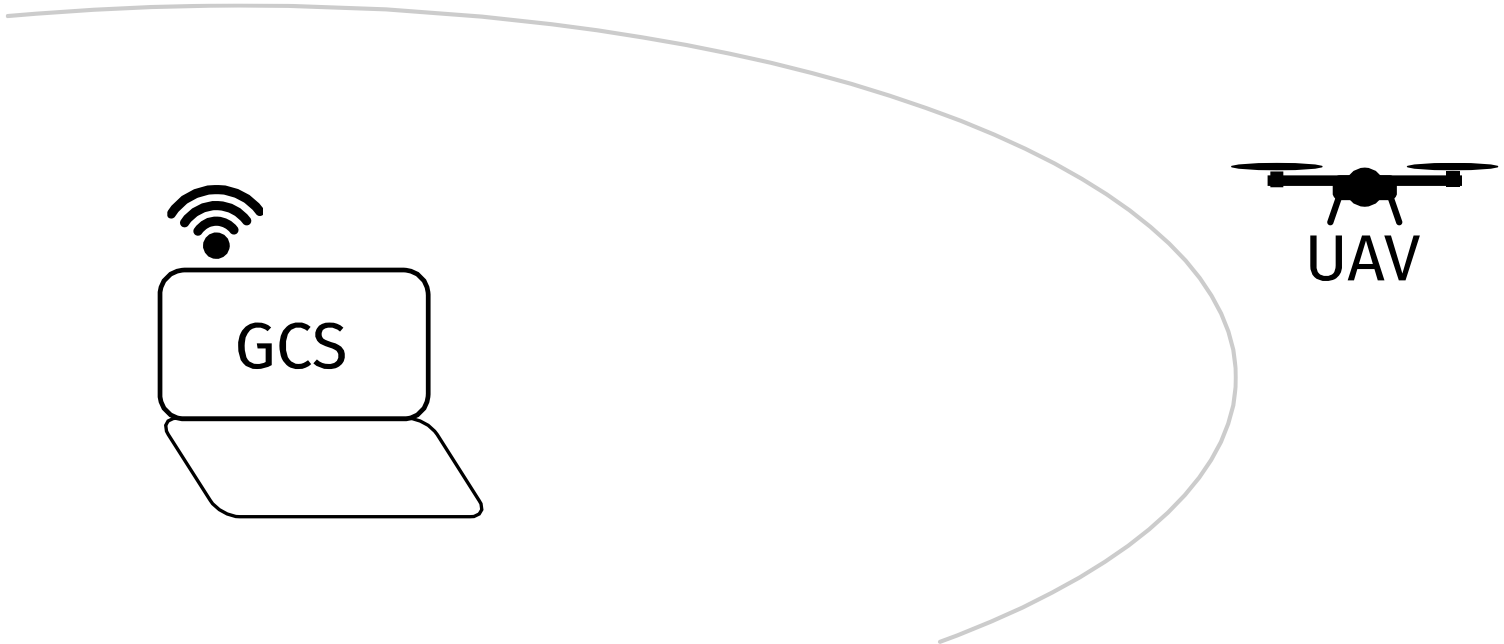
WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example



WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example

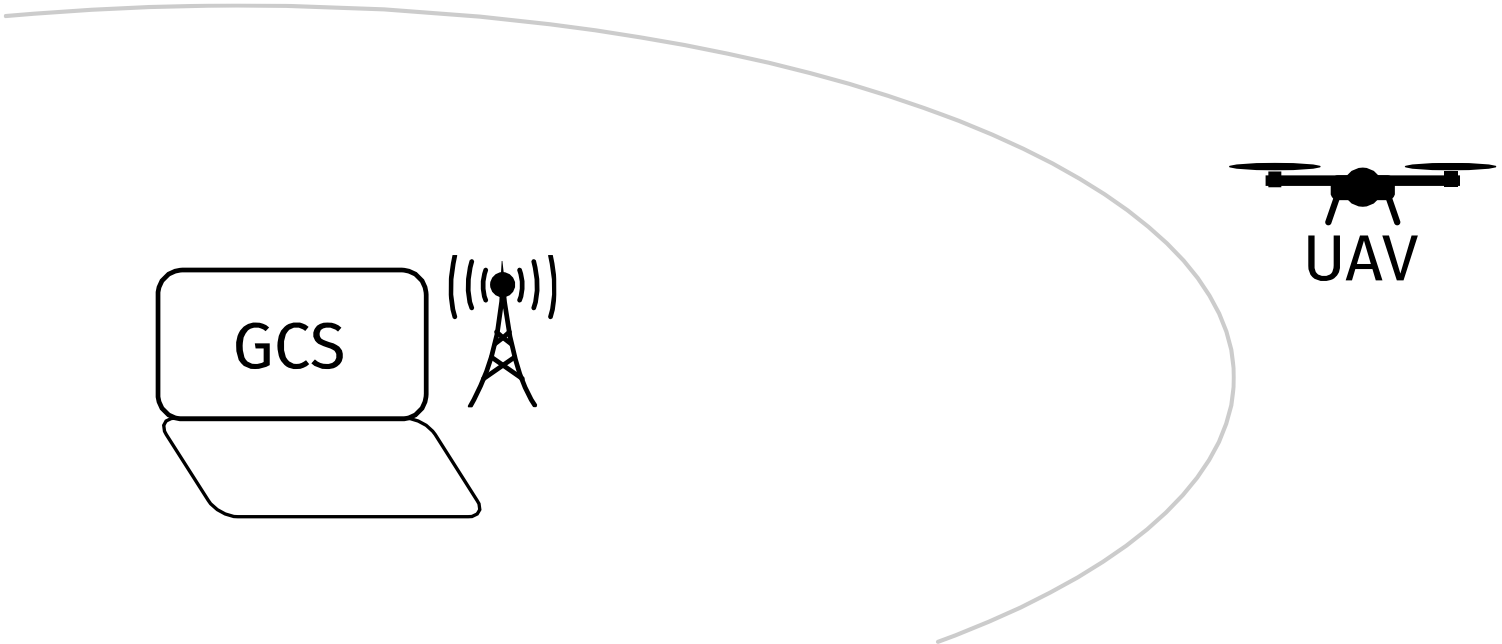


WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example

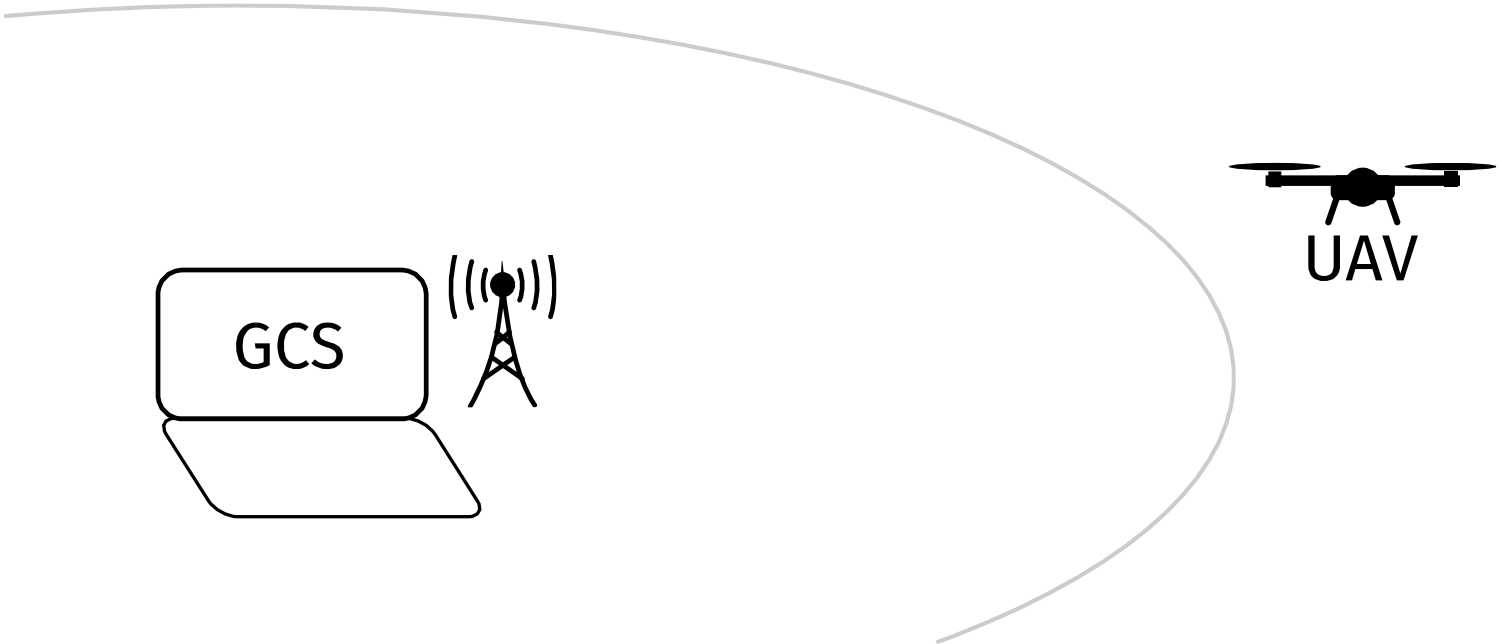
WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example



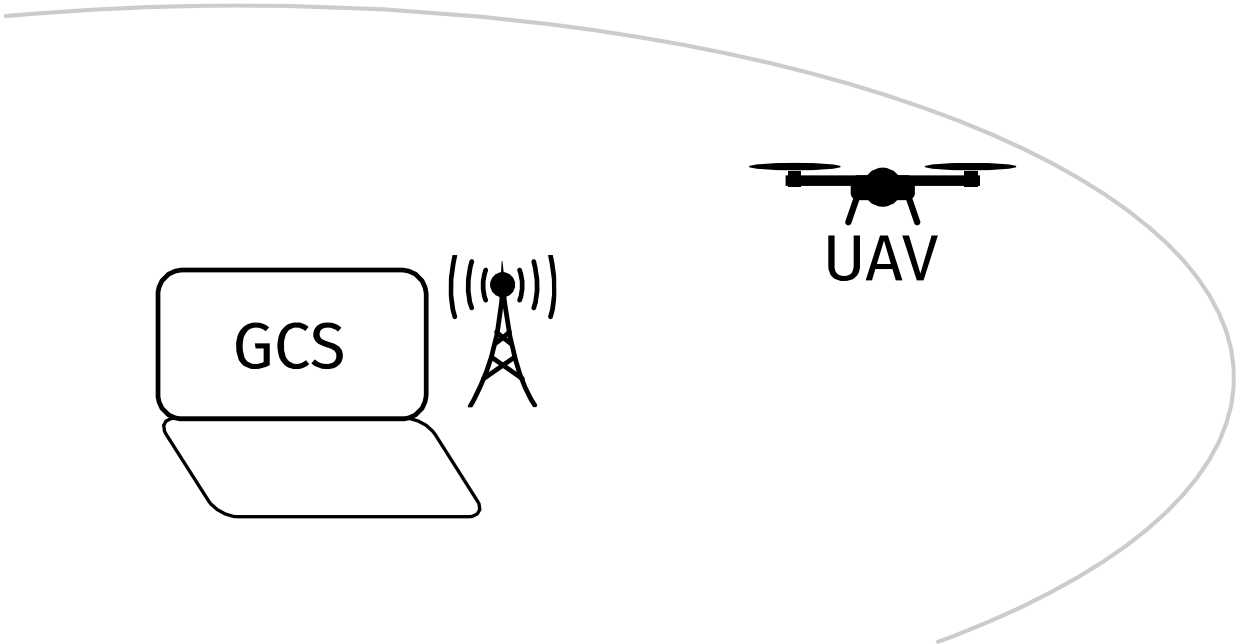
WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example



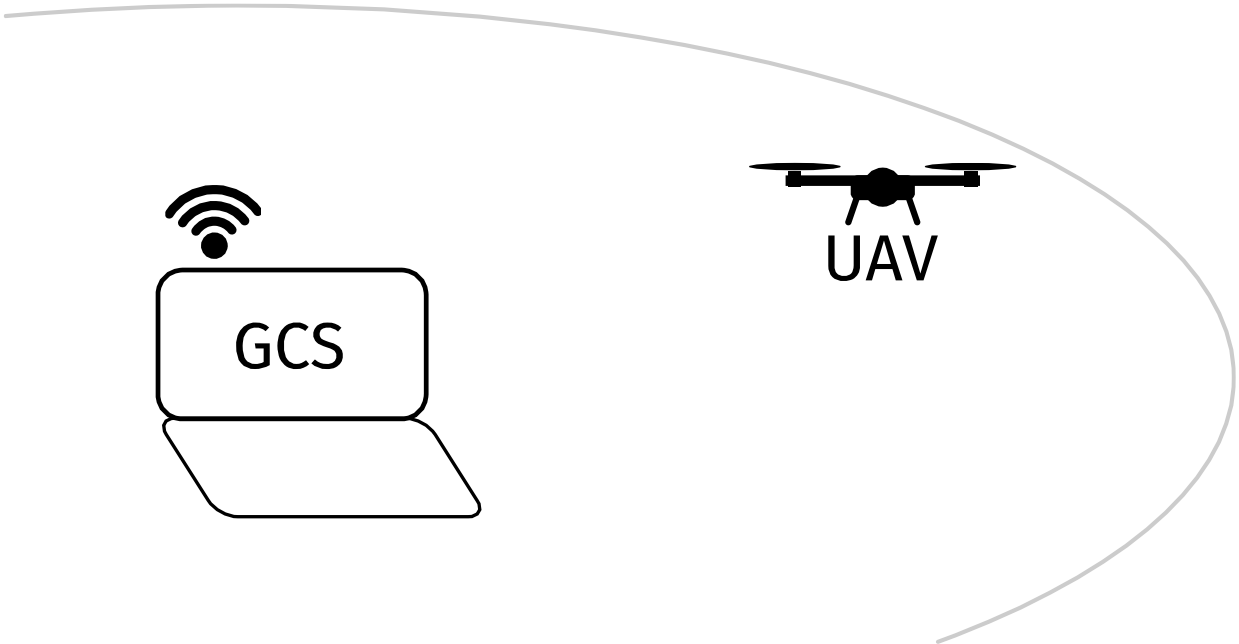
WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example



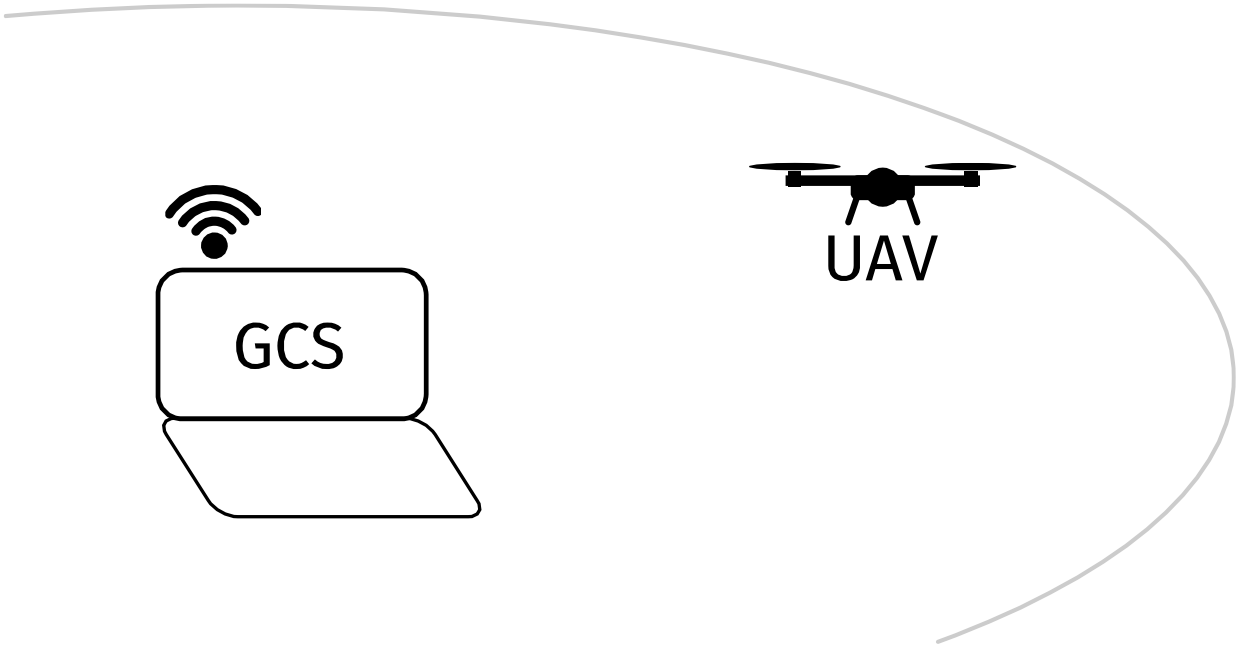
WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example



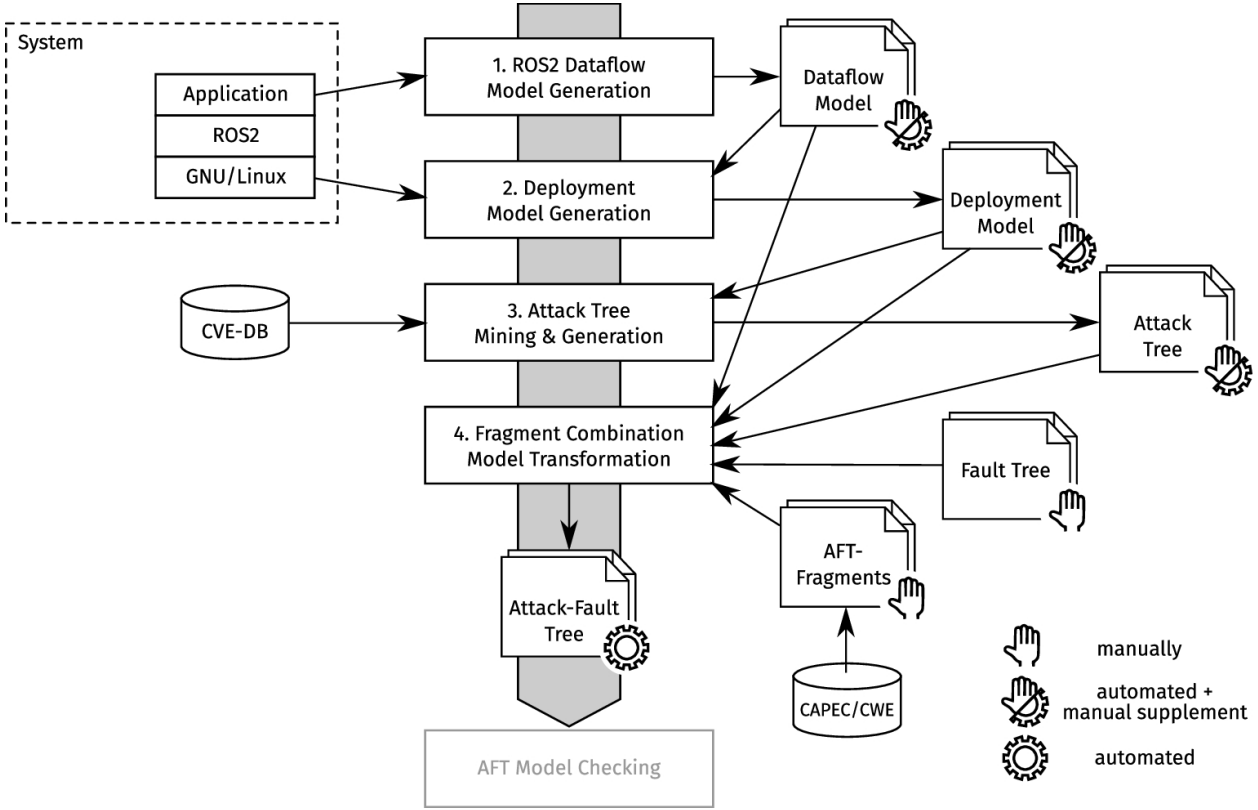
WiFi connected > Radio Connected > Return signal > WiFi connected

Running Example

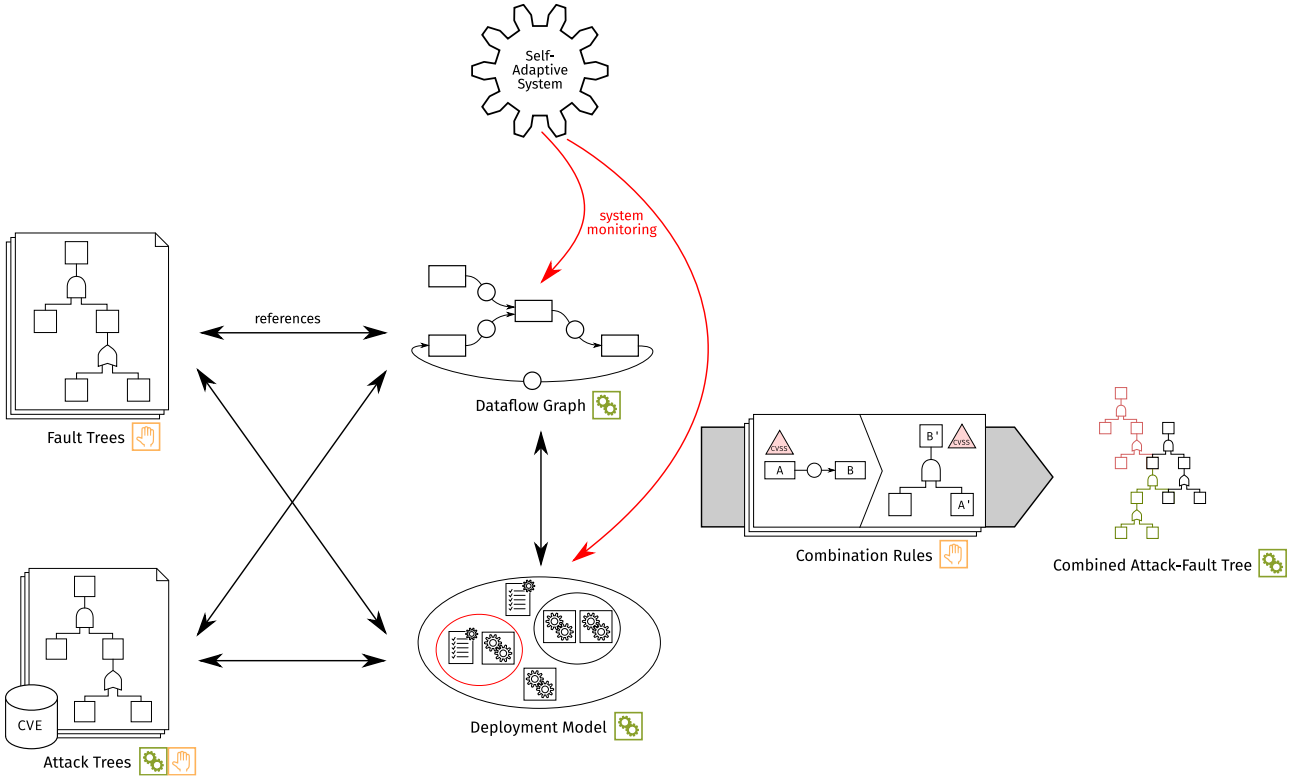


WiFi connected > Radio Connected > Return signal > WiFi connected

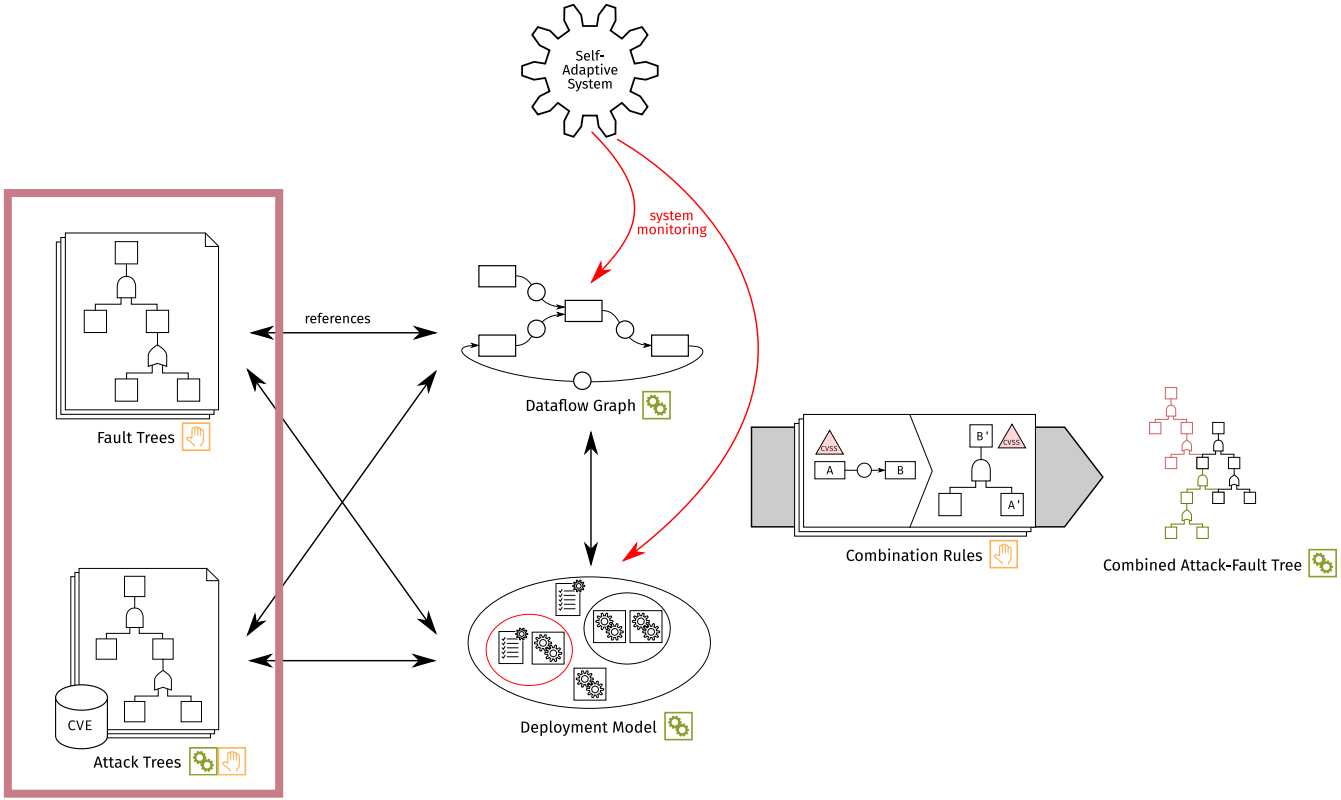
Overview of the Proposed Approach



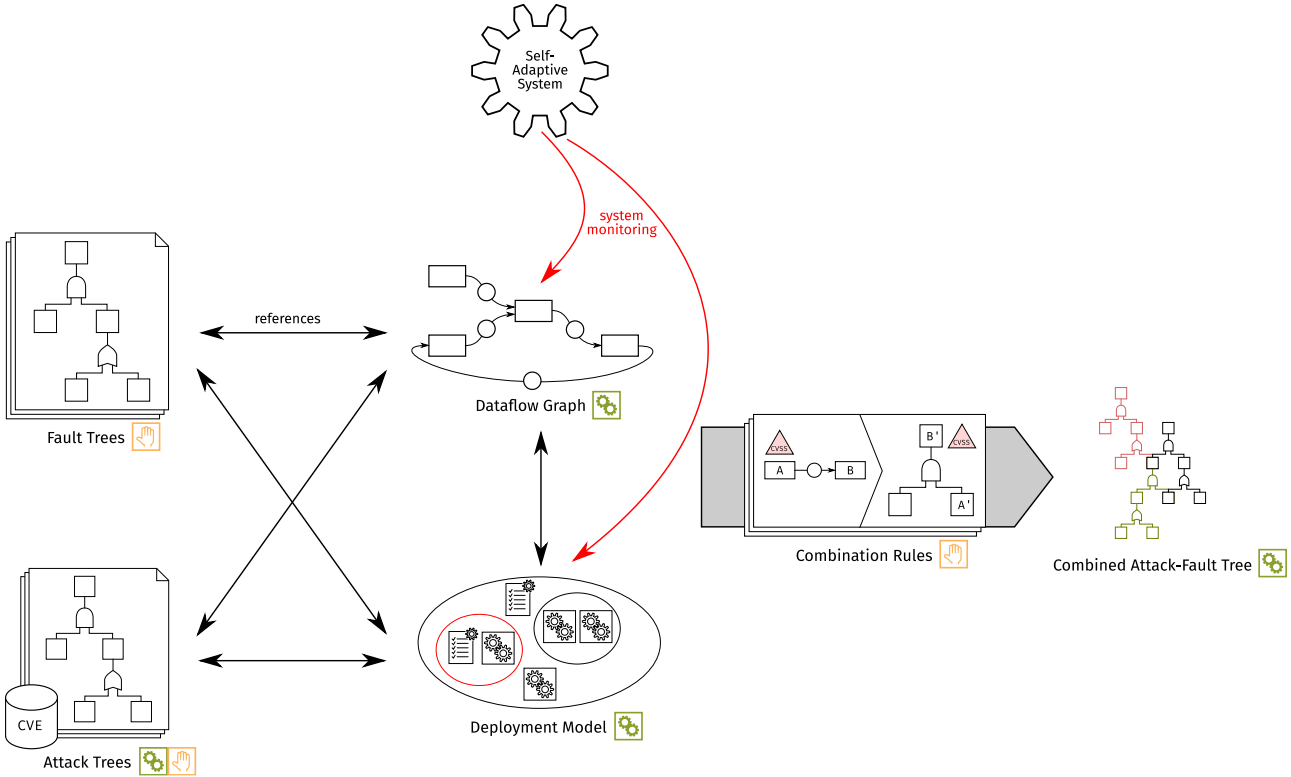
The SafeSec AFT Generation Toolchain (SAFT-GT)



The SafeSec AFT Generation Toolchain (SAFT-GT)



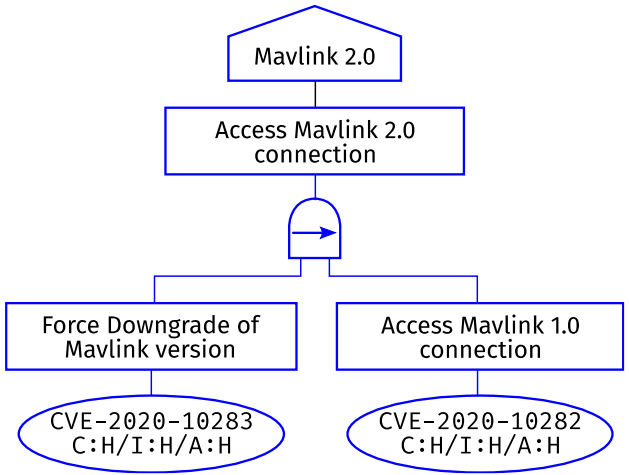
The SafeSec AFT Generation Toolchain (SAFT-GT)



SAFT-GT Models
Attack & Fault Trees

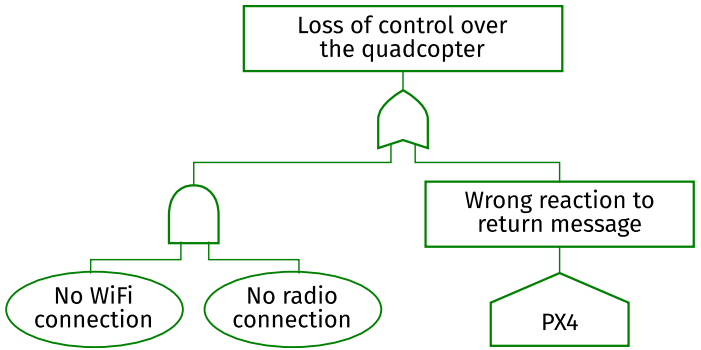
Attack Trees

- Mostly mined from CVE entries
- Often on the technical/platform level

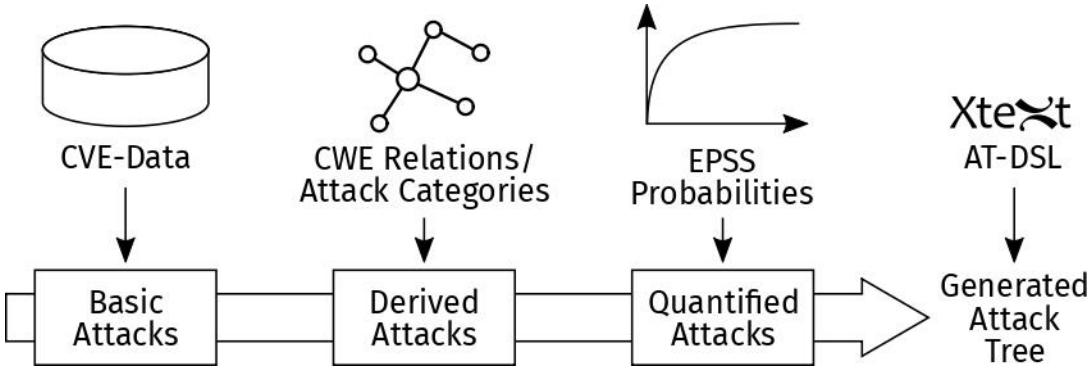


Fault Trees

- Created by Domain experts
- On the logical/dataflow level



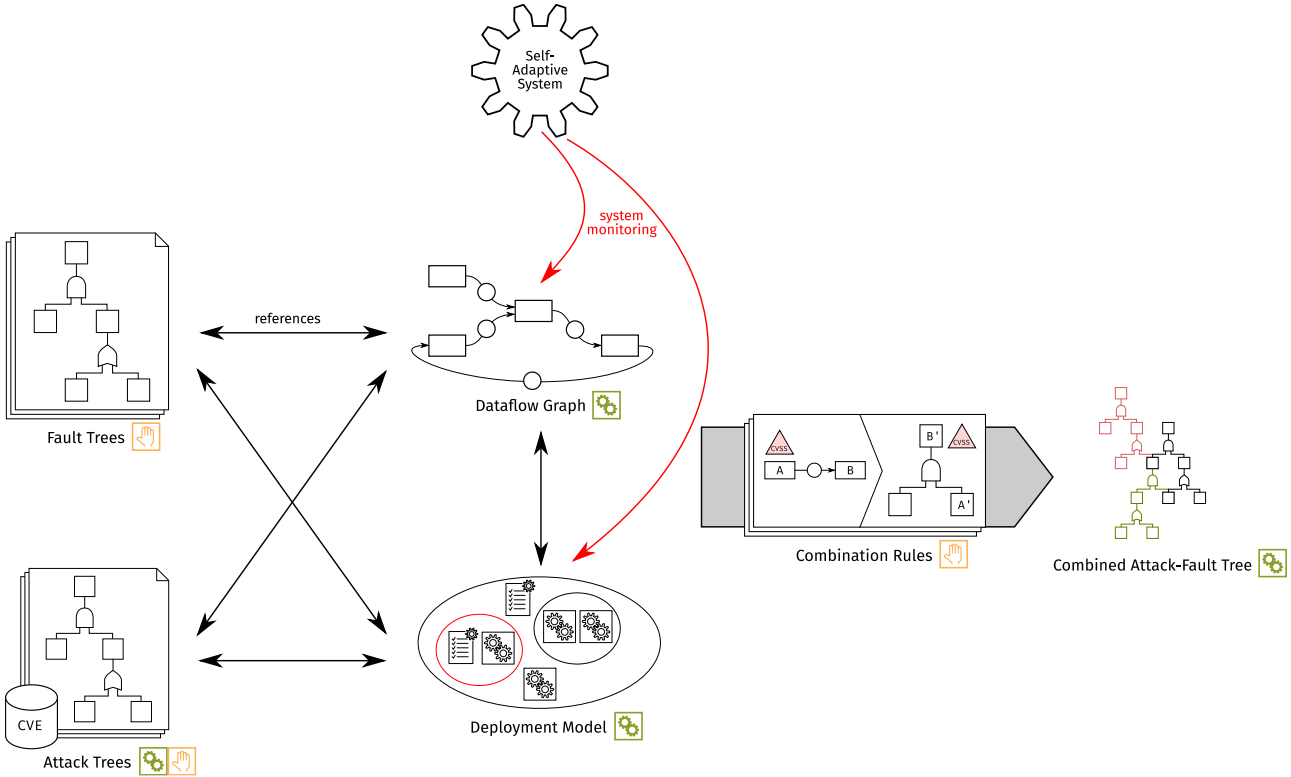
Attack Tree Generation Overview



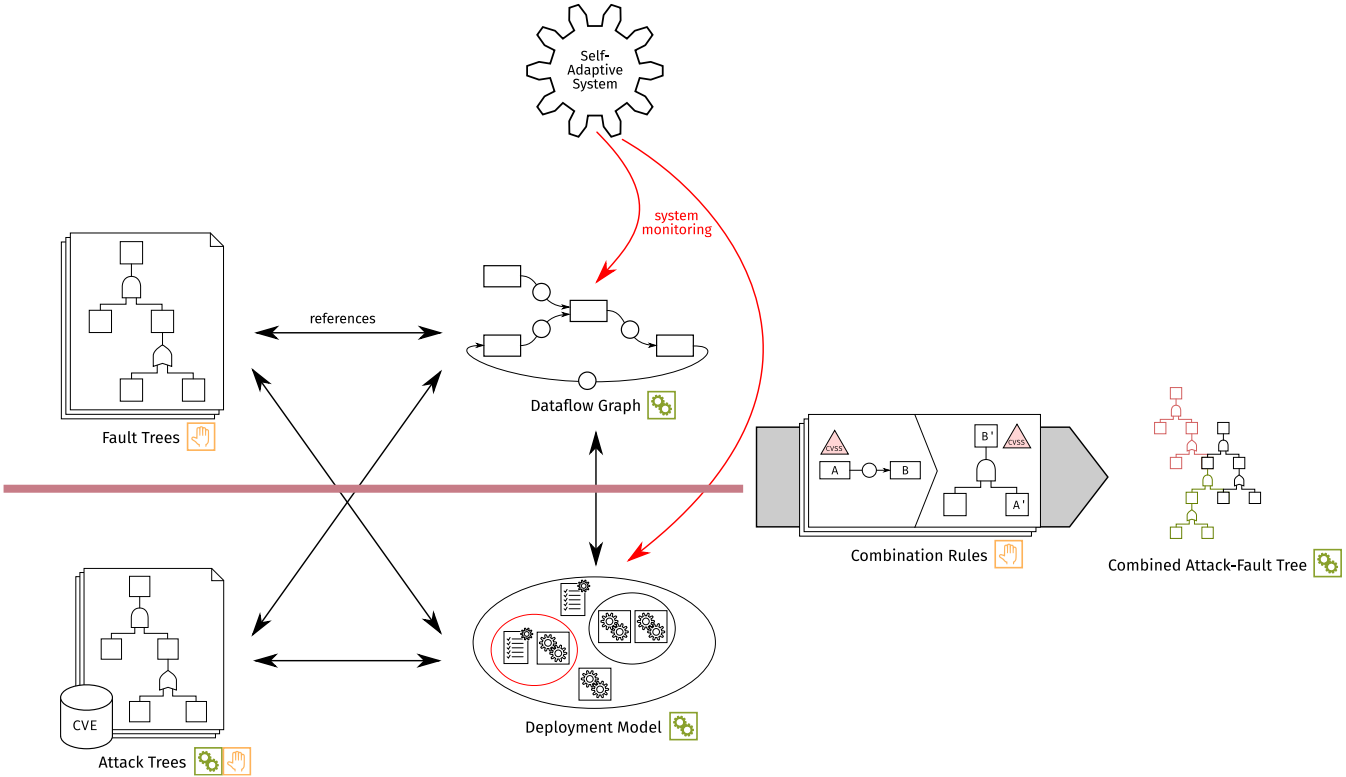
Attack Tree Generation Overview

```
1  AttackTarget:
2    "AttackTarget" (('id''='name=ID)? & ("CPE" "=" cpe=CPE)? & ("CWE" "=" cwe=CWE)?
3              &  ("CVSS" "=" cvss=CVSSVECTORList)? & ("note" "=" note=STRING)? & (("BaseScore" "=" baseScore=ScoreList)?) &
4              (("ImpactScore" "=" impactScore=ScoreList)?) & (("ExploitabilityScore" "=" exploitabilityScore=ScoreList)?))
5              "{" attackTree=AttackTree "}"
6  ;
7
8  CVSSVECTORList:
9    '[' cvssList+=CVSSVECTOR (',' cvssList+=CVSSVECTOR)* ']'
10 ;
11
12 ScoreList:
13   '[' score+=REAL (',' score+=REAL)* ']'
14 ;
15
16 AttackTree:
17   step=AttackStep| subTree=SubTree| ref=[AttackTreeSubElements]
18 ;
19
20 AttackStep:
21   "AttackStep" (name=ID)? (("description" "=" description=STRING) & (("CVE""="cve=CVE)?) & (("CVSS" "=" cvss=CVSSVECTOR)?)
22             & (("probability""="probability=REAL)?) & (("BaseScore" "=" baseScore=REAL)?) & (("ImpactScore" "=" impactScore=REAL)?)
23             & (("ExploitabilityScore" "=" exploitabilityScore=REAL)?) & (("epss" "=" epss=REAL)?) & (("note" "=" note=STRING)?))
24 ;
25
26 SubTree:
27   gate=Gate (name=ID)?  ("note" "=" note=STRING)?  "{" attackTree+=AttackTree (','attackTree+=AttackTree)* "}"
28 ;
29 Gate:
30   name='AND'| name='OR'| name='SAND'
31 ;
```

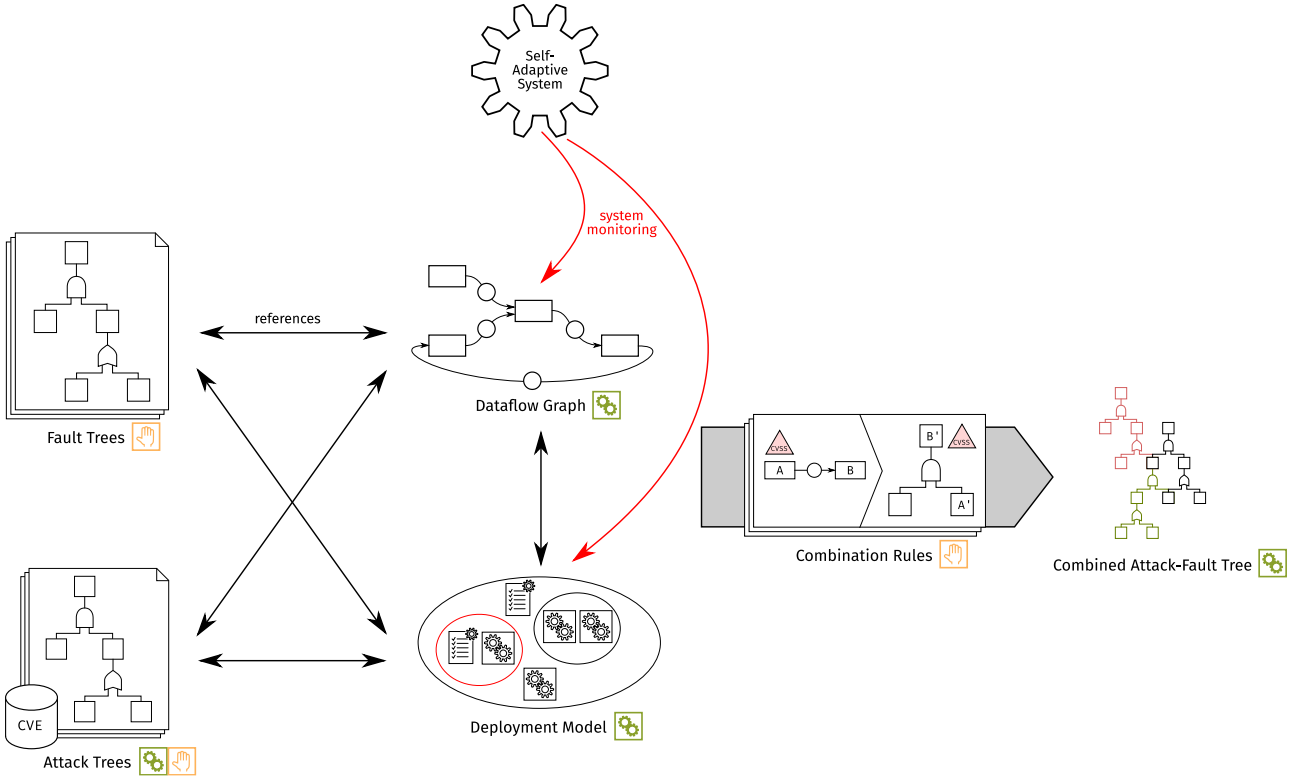
The SafeSec AFT Generation Toolchain (SAFT-GT)



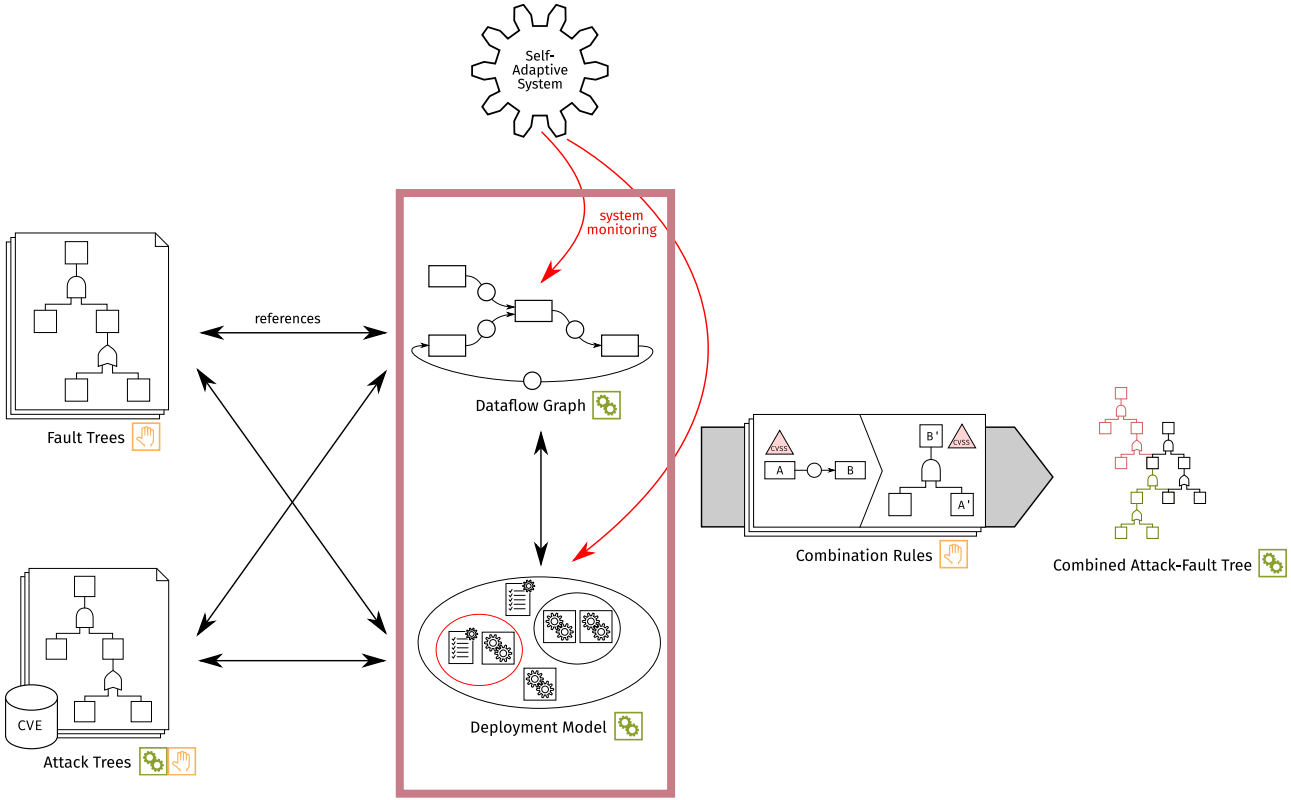
The SafeSec AFT Generation Toolchain (SAFT-GT)



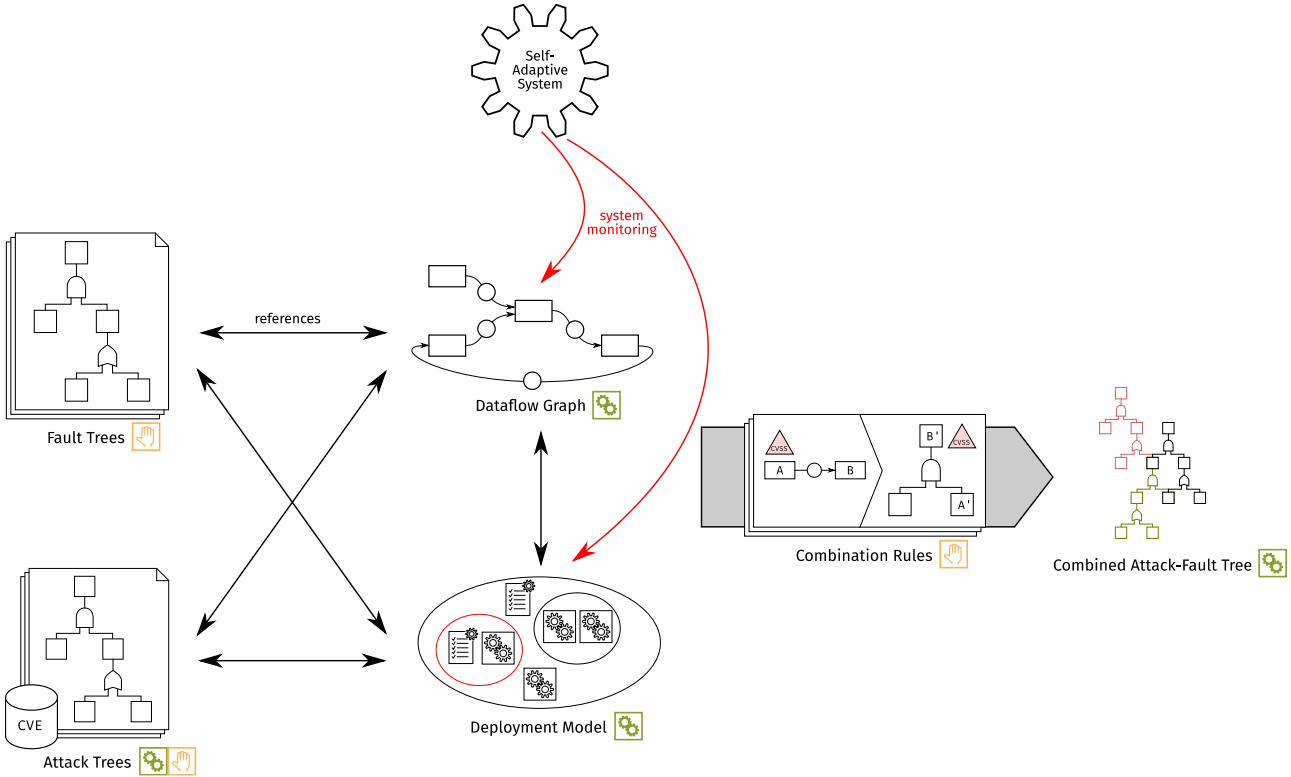
The SafeSec AFT Generation Toolchain (SAFT-GT)



The SafeSec AFT Generation Toolchain (SAFT-GT)



The SafeSec AFT Generation Toolchain (SAFT-GT)



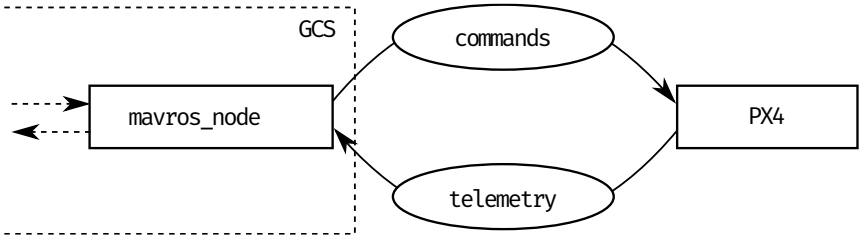
SAFT-GT Models
Dataflow Model & Deployment Model

Dataflow Model

- Simple, only components and channels
- Logical view, independent from underlying realization/implementation

Deployment Model

- Realization of components and channels
- Simple set-based representation, can be refined as needed

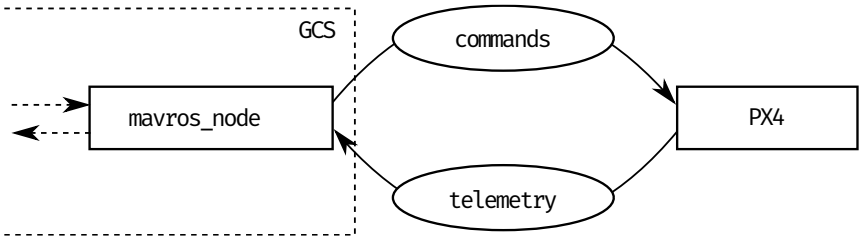


PX4:Component → {Mavlink2.0, pixhawk, ...}
Mavlink2.0:Protocol → {MavlinkLib, UDP, UART, ...}
commands:Channel → {Mavlink2.0, WiFi}
WiFi:Platform → {IEEE 802.11n, WPA2, UDP, TCP/IP, ...}
Radio:Platform → {UART}

SAFT-GT Models
Dataflow Model & Deployment Model

Dataflow Model

- Simple, only components and channels
- Logical view, independent from underlying realization/implementation

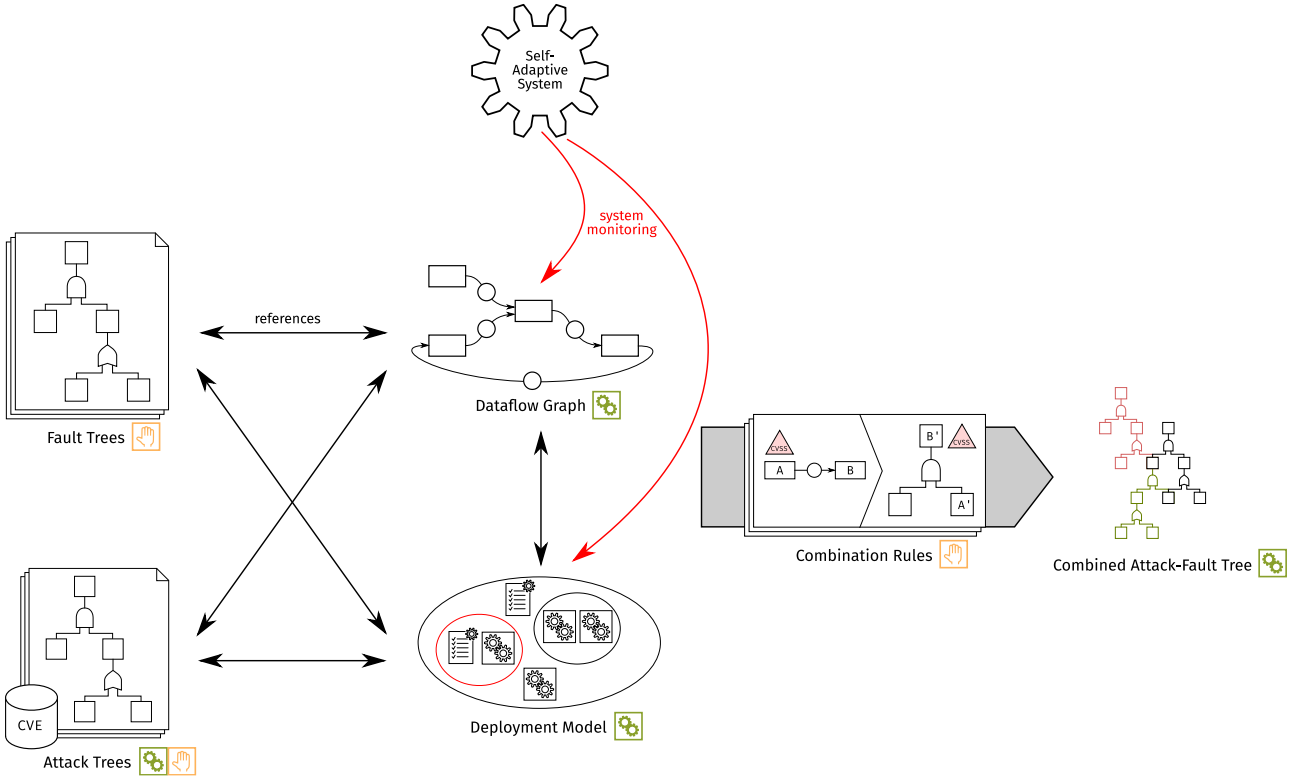


Deployment Model

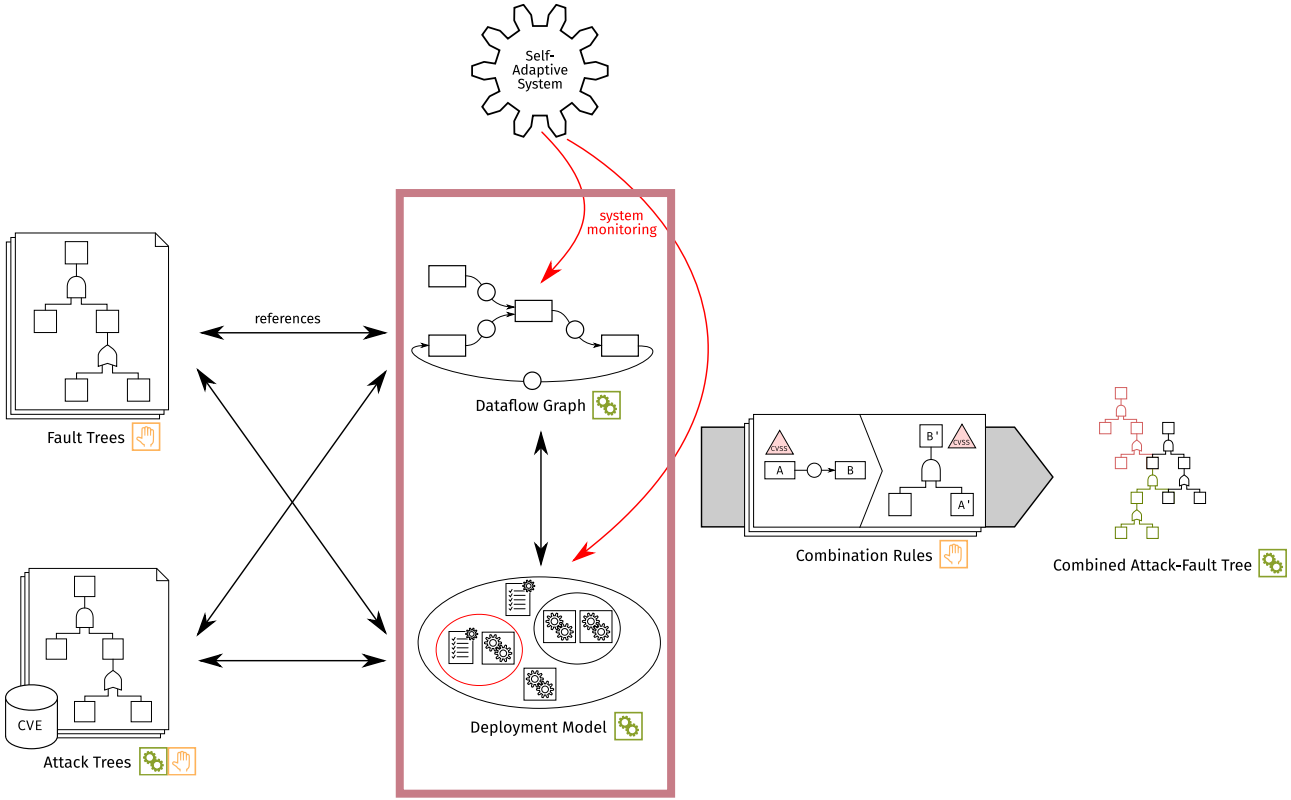
- Realization of components and channels
- Simple set-based representation, can be refined as needed

PX4:Component → {Mavlink2.0, pixhawk, ...}
Mavlink2.0:Protocol → {MavlinkLib, UDP, UART, ...}
commands:Channel → {Mavlink2.0, Radio}
WiFi:Platform → {IEEE 802.11n, WPA2, UDP, TCP/IP, ...}
Radio:Platform → {UART}

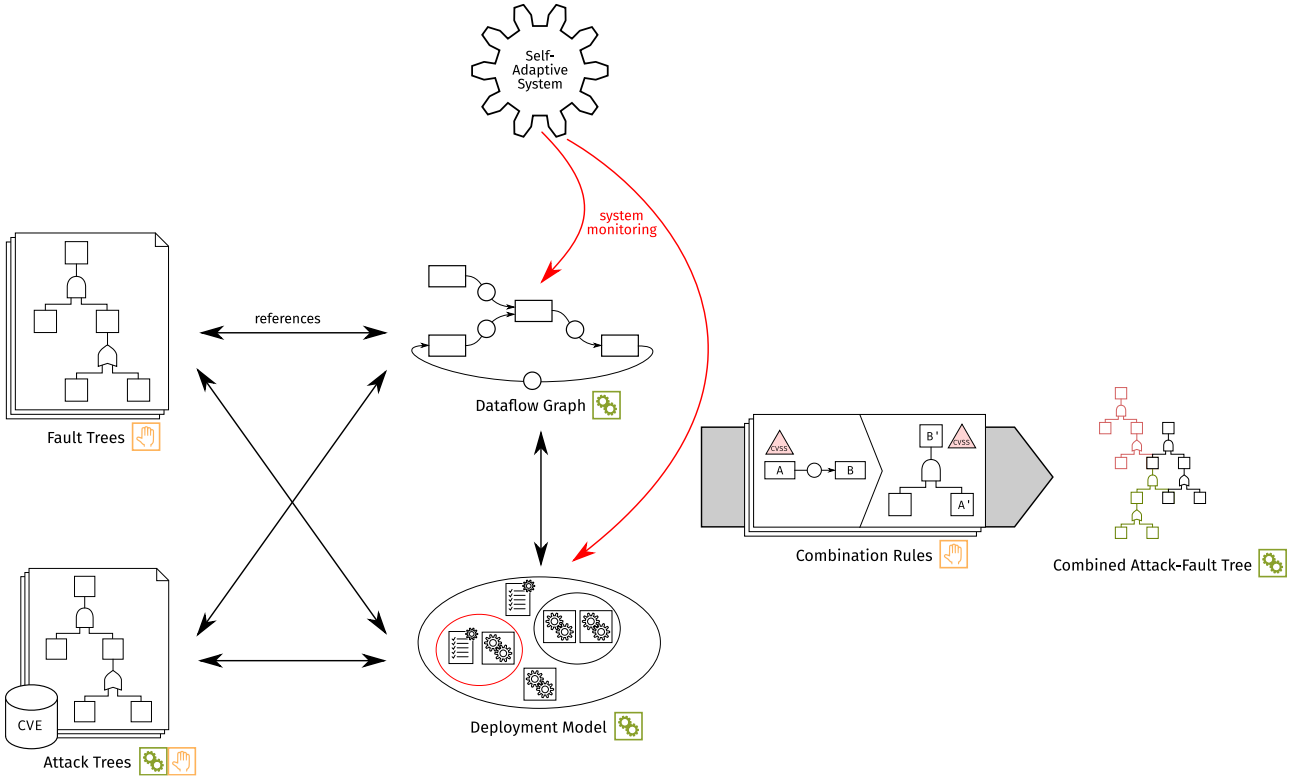
The SafeSec AFT Generation Toolchain (SAFT-GT)



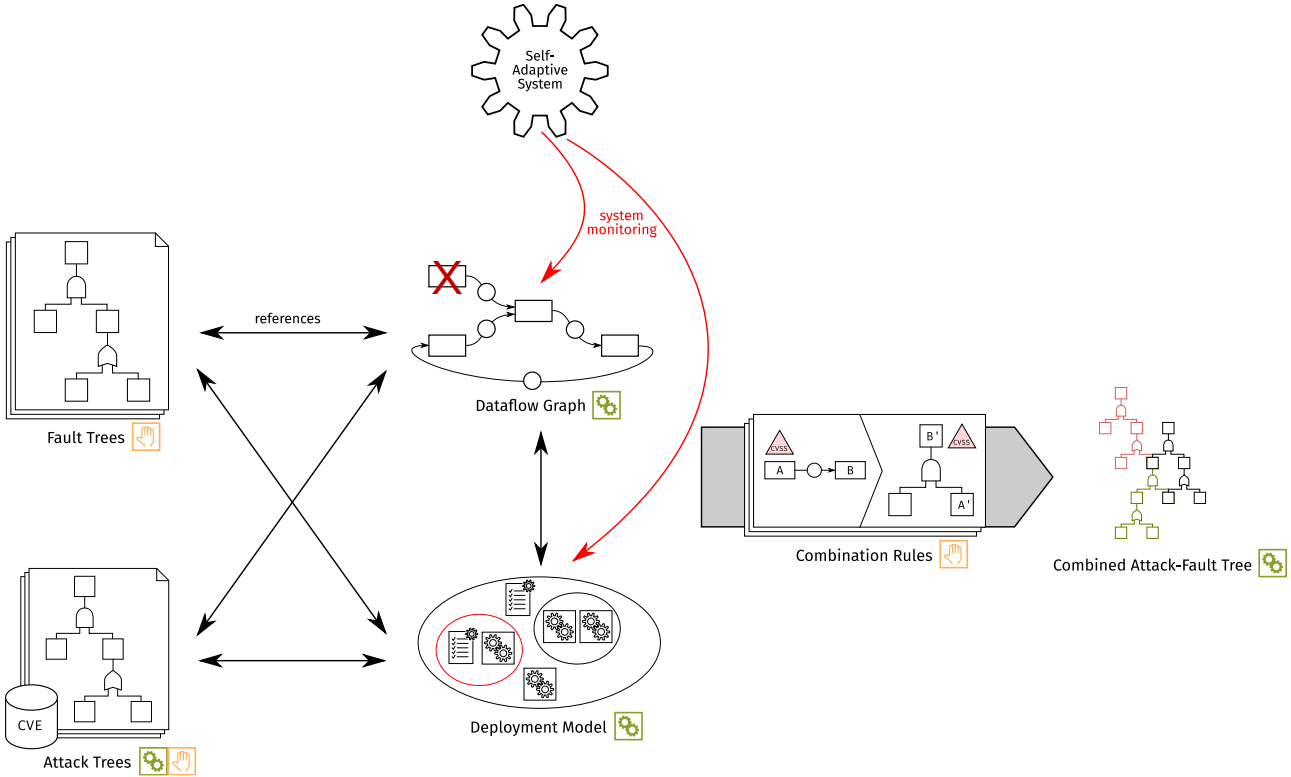
The SafeSec AFT Generation Toolchain (SAFT-GT)



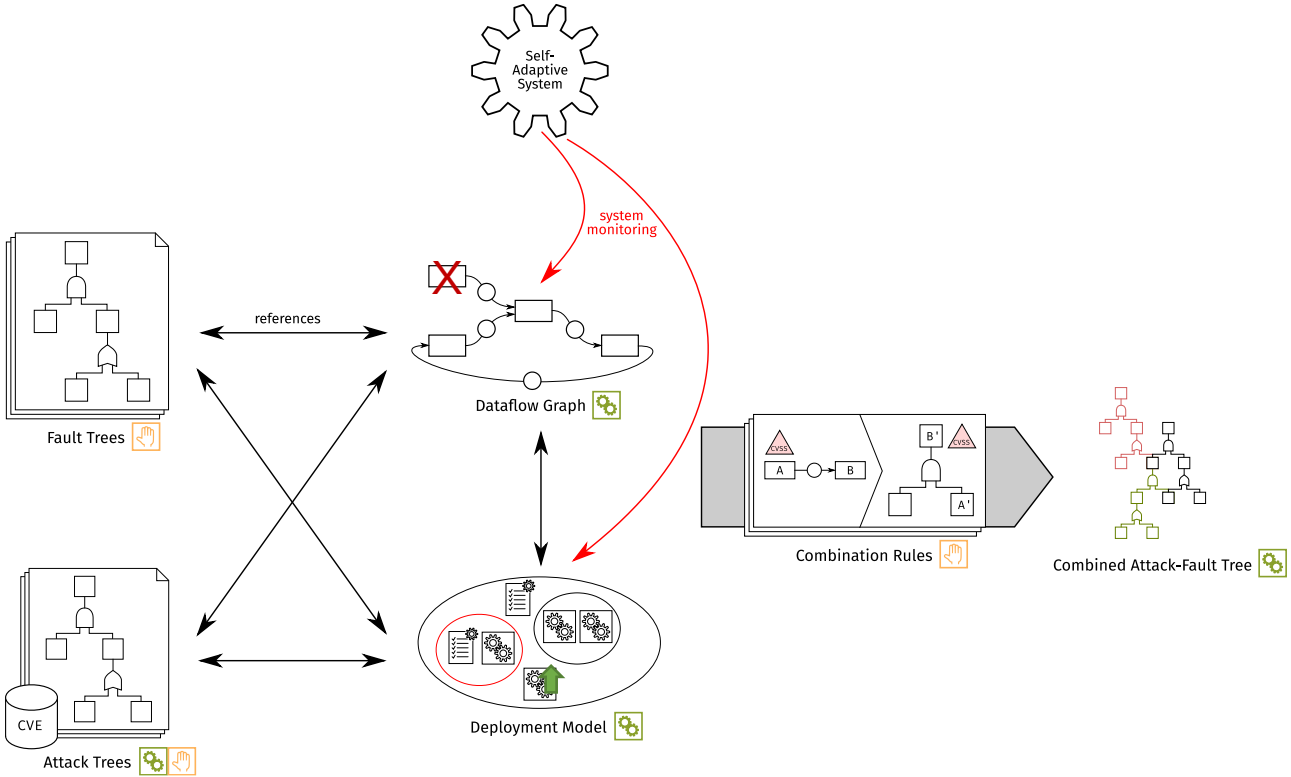
The SafeSec AFT Generation Toolchain (SAFT-GT)



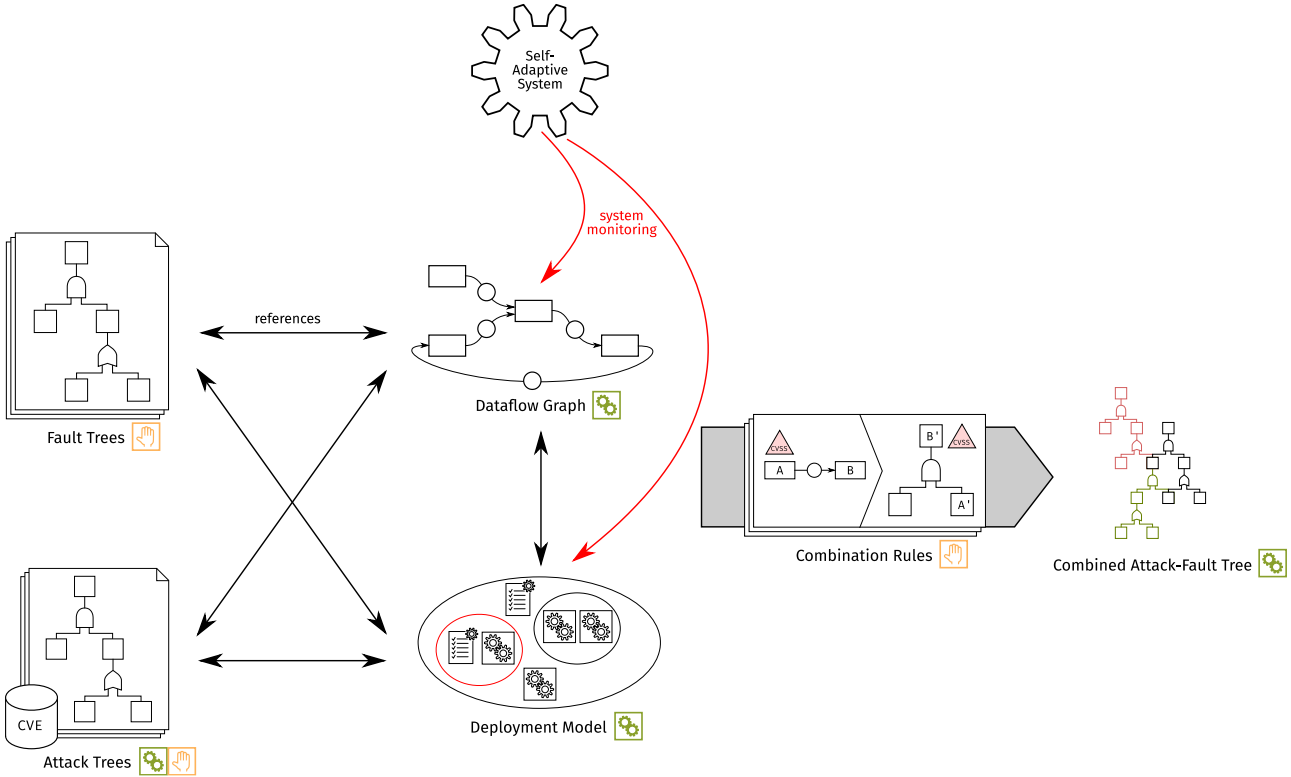
The SafeSec AFT Generation Toolchain (SAFT-GT)



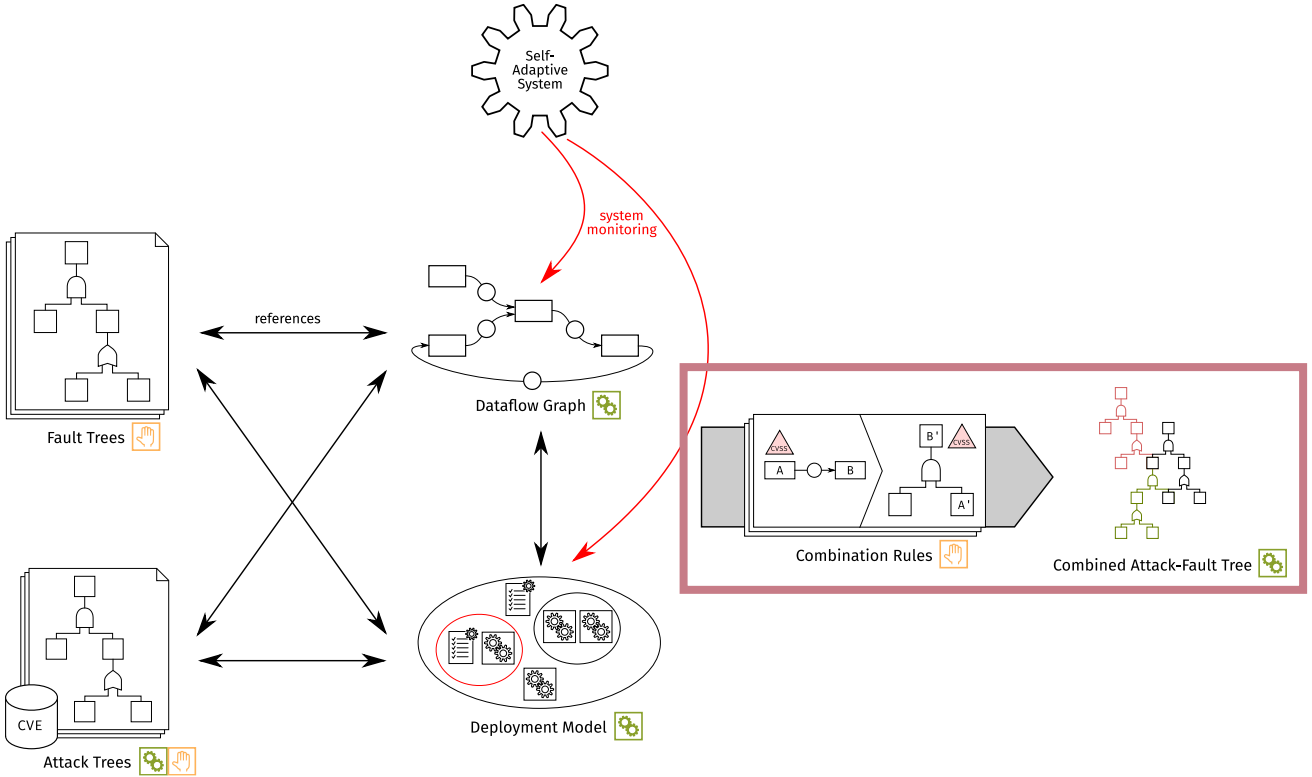
The SafeSec AFT Generation Toolchain (SAFT-GT)



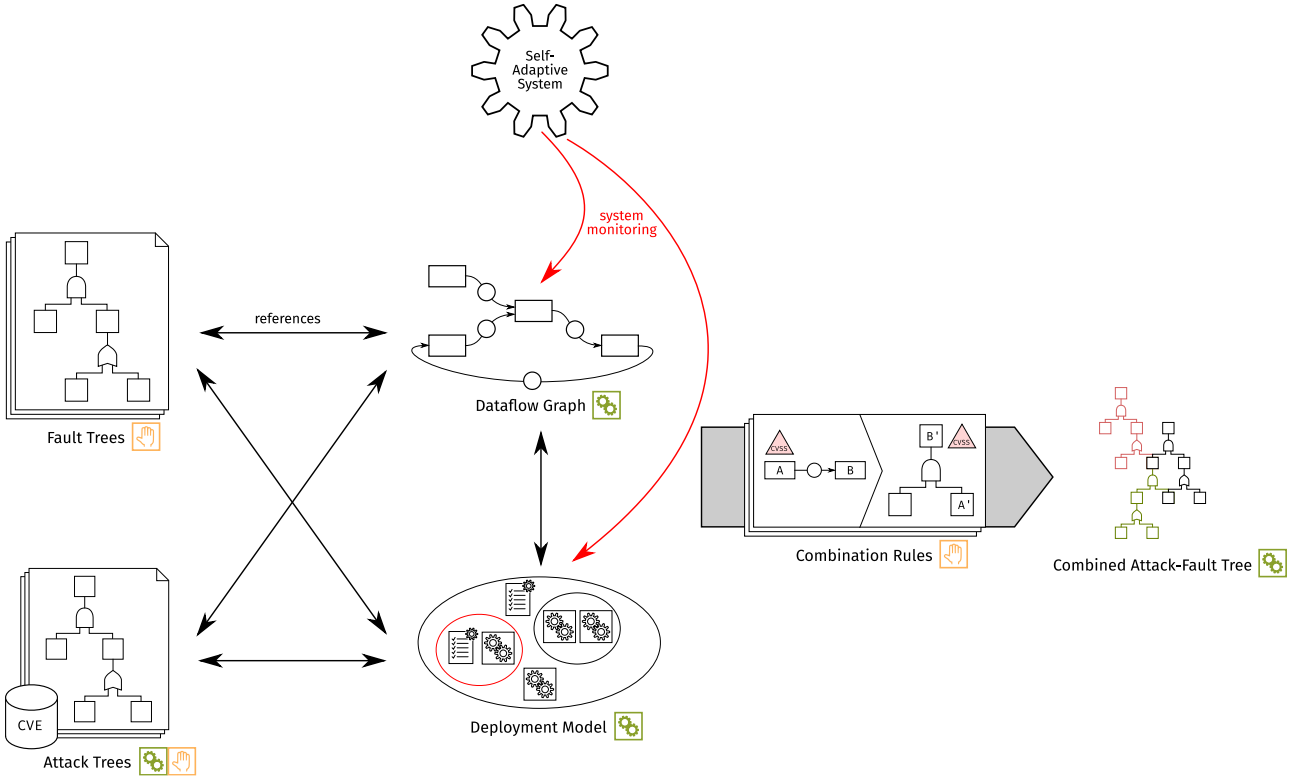
The SafeSec AFT Generation Toolchain (SAFT-GT)



The SafeSec AFT Generation Toolchain (SAFT-GT)

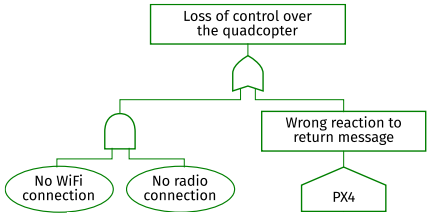


The SafeSec AFT Generation Toolchain (SAFT-GT)



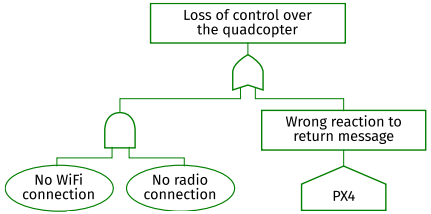
SAFT-GT
Translation Rules & AFT Generation

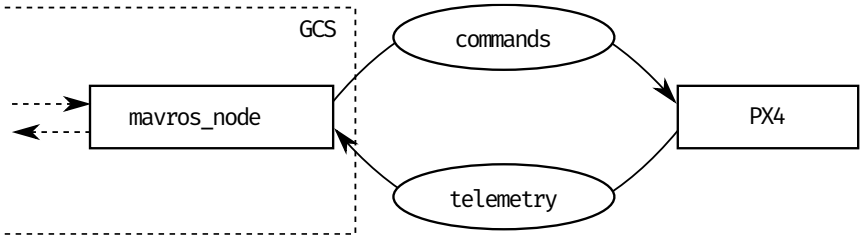
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments



SAFT-GT
Translation Rules & AFT Generation

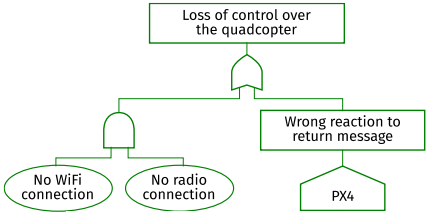
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments

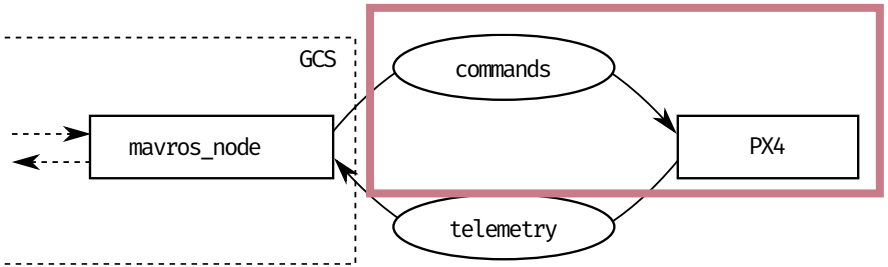




SAFT-GT
Translation Rules & AFT Generation

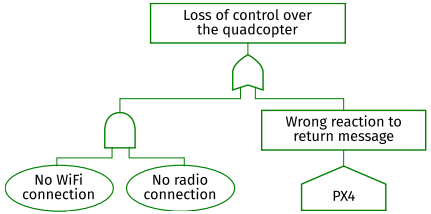
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments

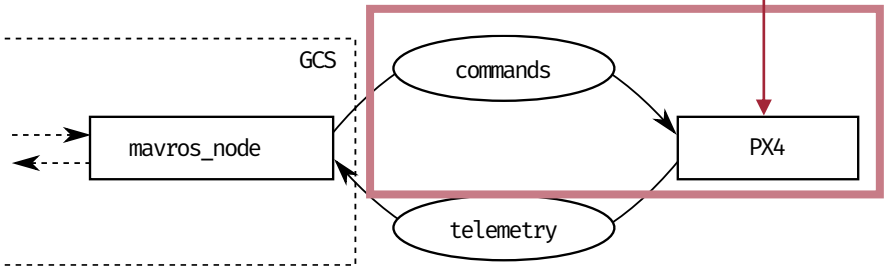




SAFT-GT
Translation Rules & AFT Generation

- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments

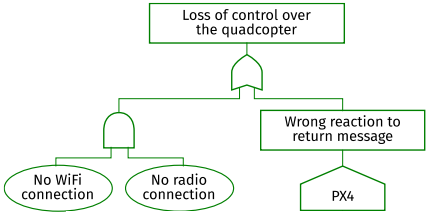


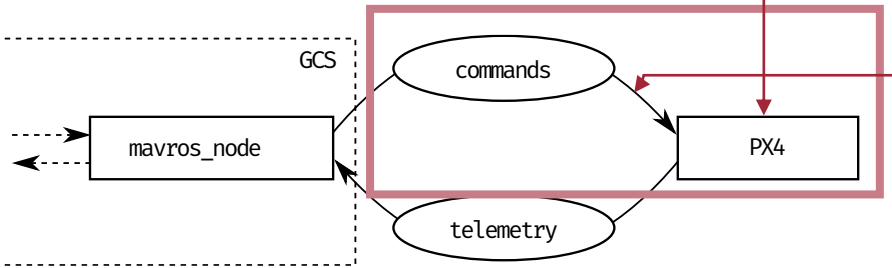



```
//1. Precondition: A component must be referenced
if(not self.oclIsTypeOf(dataflow::Component)){
    return patternList;
};
var comp:dataflow::Component:=self.oclAsType(dataflow::Component);
```

SAFT-GT
Translation Rules & AFT Generation

- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments

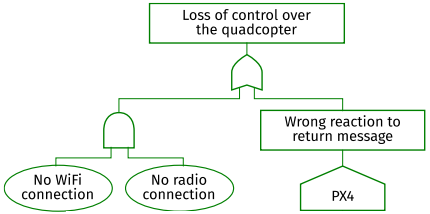


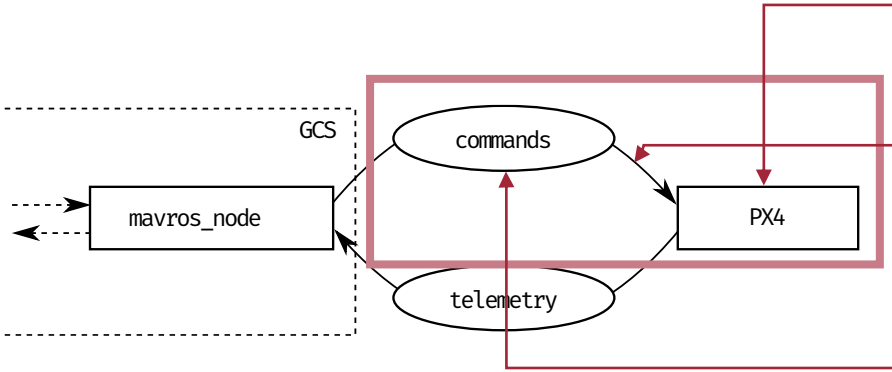



```
//1. Precondition: A component must be referenced
if(not self.oclIsTypeOf(dataflow::Component)){
    return patternList;
};
var comp:dataflow::Component:=self.oclAsType(dataflow::Component);


//2. Precondition: Component has one incoming channel
var inportList:=comp.elements->select(ele|ele.oclIsTypeOf(dataflow::InPort));
if(inportList->oclIsUndefined() or inportList->isEmpty()){
    return patternList;
};
var incomingChannelList:=sourceInDF.rootObjects()[dataflow::Model]
    ->allSubobjectsOfKind(dataflow::Connection)->flatten()
    ->select(con|con.port.oclIsTypeOf(dataflow::InPort) and
        inportList->includes(con.port)).channel;
```

SAFT-GT
Translation Rules & AFT Generation

- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments

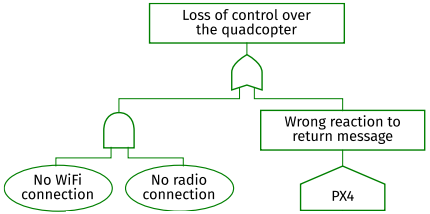


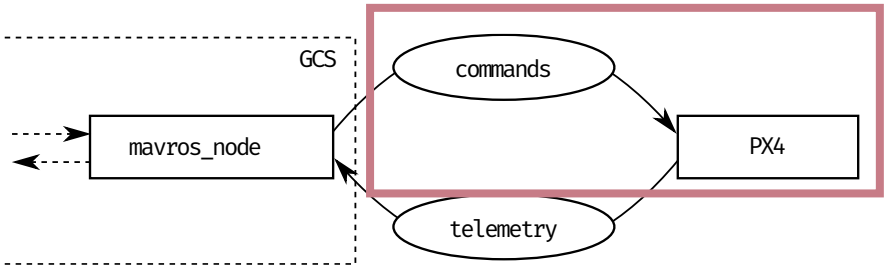



```
//1. Precondition: A component must be referenced
if(not self.oclIsTypeOf(dataflow::Component)){
    return patternList;
};
var comp:dataflow::Component:=self.oclAsType(dataflow::Component);

//2. Precondition: Component has one incoming channel
var inportList:=comp.elements->select(ele|ele.oclIsTypeOf(dataflow::InPort));
if(inportList->oclIsUndefined() or inportList->isEmpty()){
    return patternList;
};
var incomingChannelList:=sourceInDF.rootObjects()[dataflow::Model]
    ->allSubobjectsOfKind(dataflow::Connection)->flatten()
    ->select(con|con.port.oclIsTypeOf(dataflow::InPort) and
        inportList->includes(con.port)).channel;

//3. Precondition: Channel uses something related to an attack
var channelDefinitionList:=sourceInD.rootObjects()[deployment::Model]
    ->allSubobjectsOfKind(deployment::Element)->flatten()
    ->select(ele|ele.element.oclIsTypeOf(deployment::RefChannel) and incomingChannelList
        ->includes(ele.element.oclAsType(deployment::RefChannel).linkedDataflowChannel))
    ->select(ele|ele.checkForAffectedChannel())->asSequence();
```

SAFT-GT
Translation Rules & AFT Generation

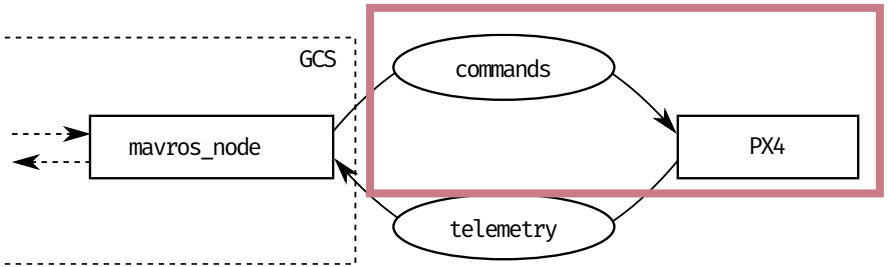
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments

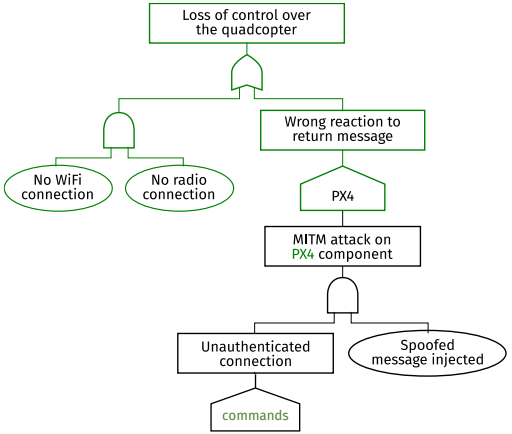




SAFT-GT
Translation Rules & AFT Generation

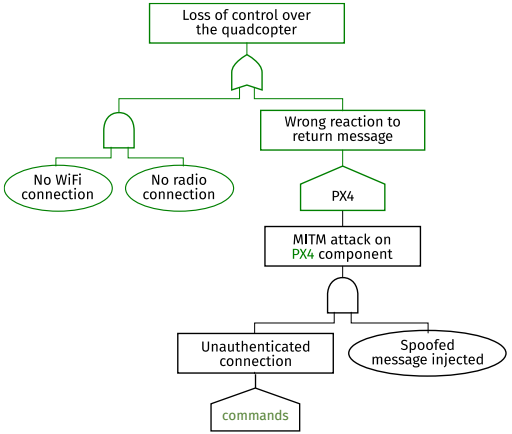
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments





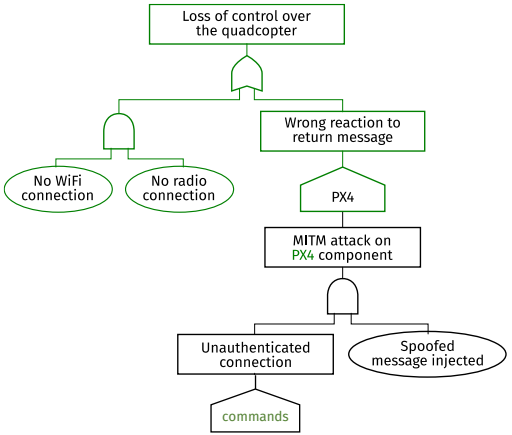
SAFT-GT
Translation Rules & AFT Generation

- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments



SAFT-GT
Translation Rules & AFT Generation

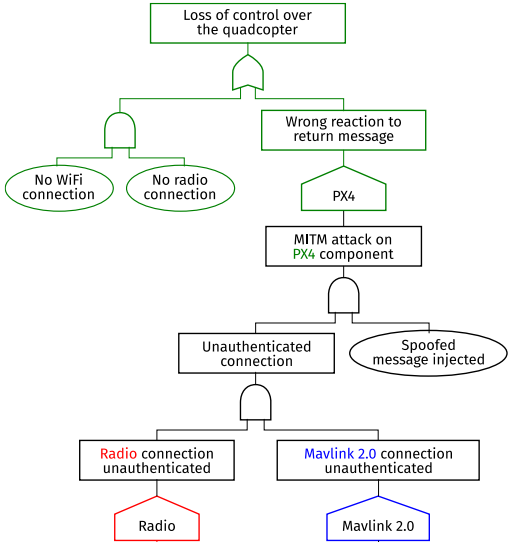
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments



commands:Channel → {Mavlink2.0, Radio}

SAFT-GT
Translation Rules & AFT Generation

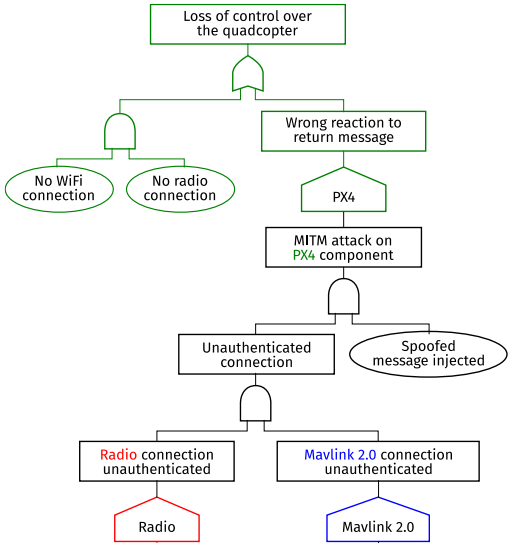
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments



commands:Channel → {Mavlink2.0, Radio}

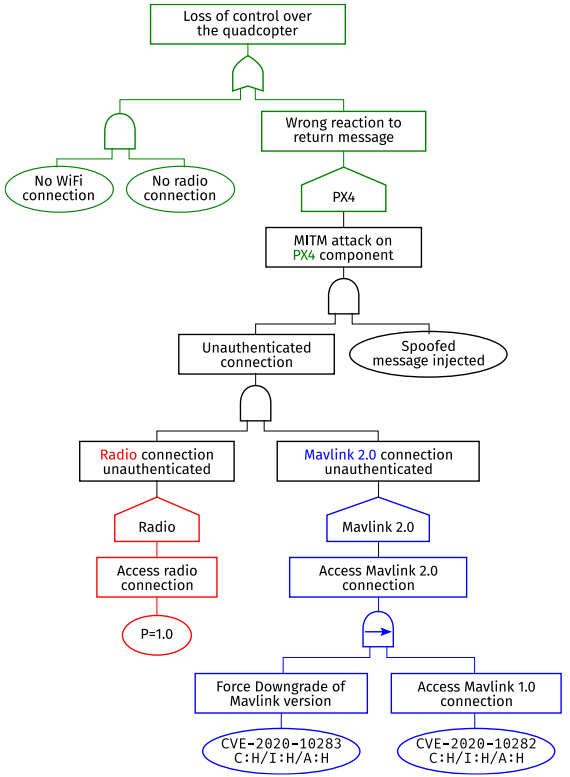
SAFT-GT
Translation Rules & AFT Generation

- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments



SAFT-GT
Translation Rules & AFT Generation

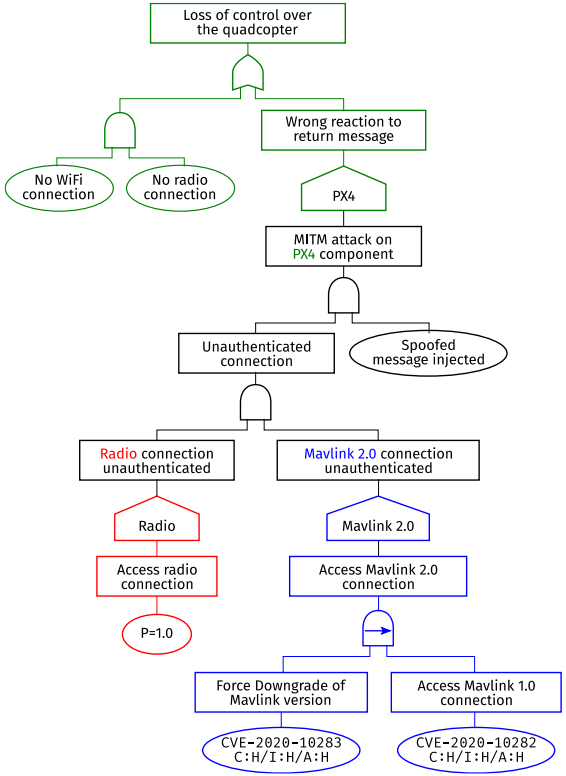
- Attack-Fault-Trees are iteratively grown
- Patterns are derived from CAPEC
- Rules transform Dataflow & Deployment models to AFT fragments



Compatibility Constraints

Rule application needs to be constrained

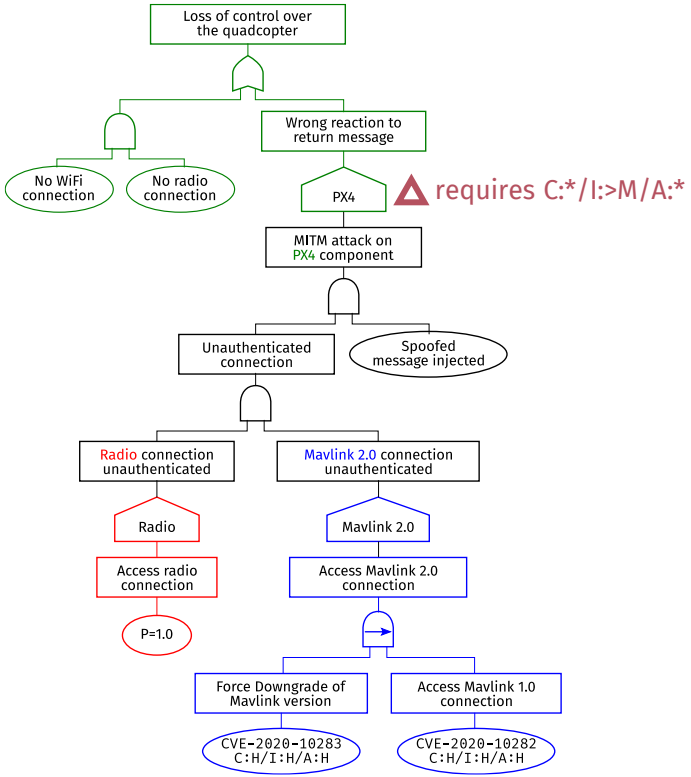
- Entity type: e.g. Library, Component, Channel…
- Name/CPE: Attacks reference CPE, Fault Trees reference dataflow elements by name
- CVSS impact metric: a fault needs a minimal impact on e.g. availability to trigger
- Possibly additional attack classification/taxonomy



Compatibility Constraints

Rule application needs to be constrained

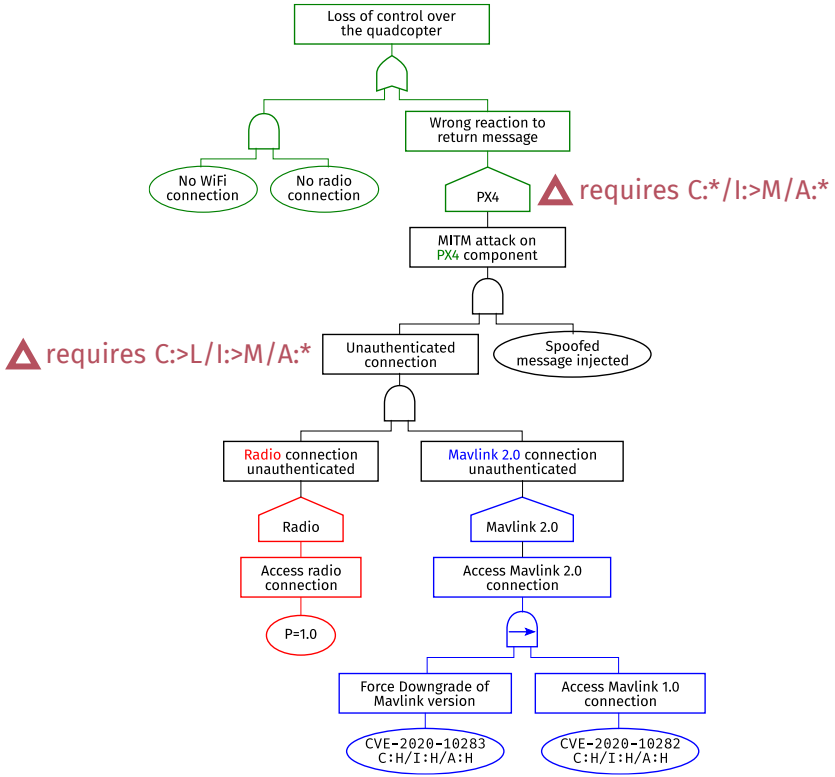
- Entity type: e.g. Library, Component, Channel…
- Name/CPE: Attacks reference CPE, Fault Trees reference dataflow elements by name
- CVSS impact metric: a fault needs a minimal impact on e.g. availability to trigger
- Possibly additional attack classification/taxonomy



Compatibility Constraints

Rule application needs to be constrained

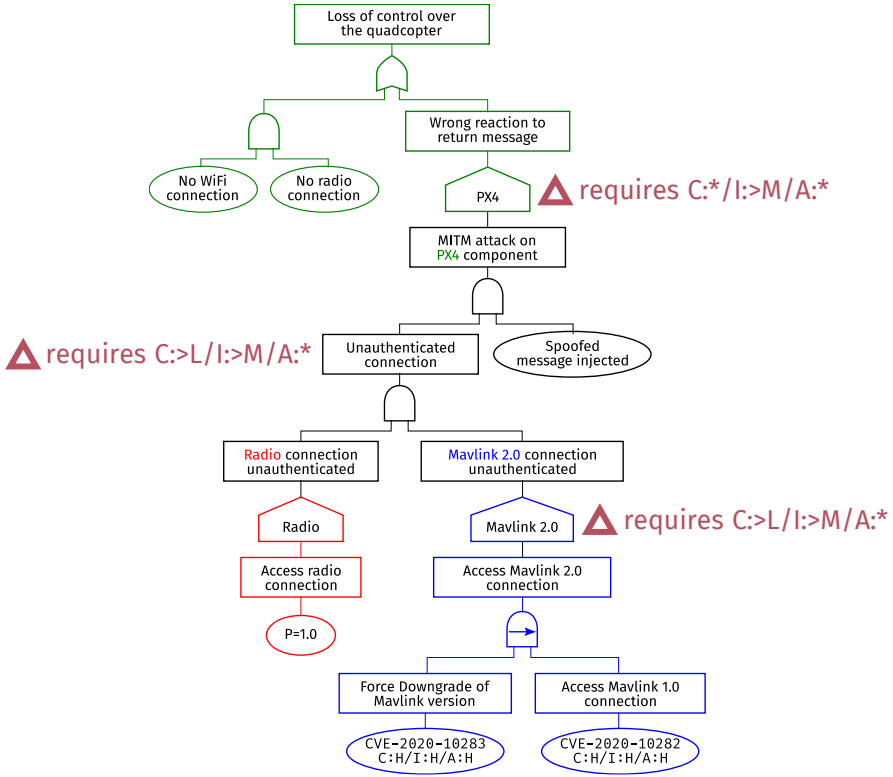
- Entity type: e.g. Library, Component, Channel…
- Name/CPE: Attacks reference CPE, Fault Trees reference dataflow elements by name
- CVSS impact metric: a fault needs a minimal impact on e.g. availability to trigger
- Possibly additional attack classification/taxonomy



Compatibility Constraints

Rule application needs to be constrained

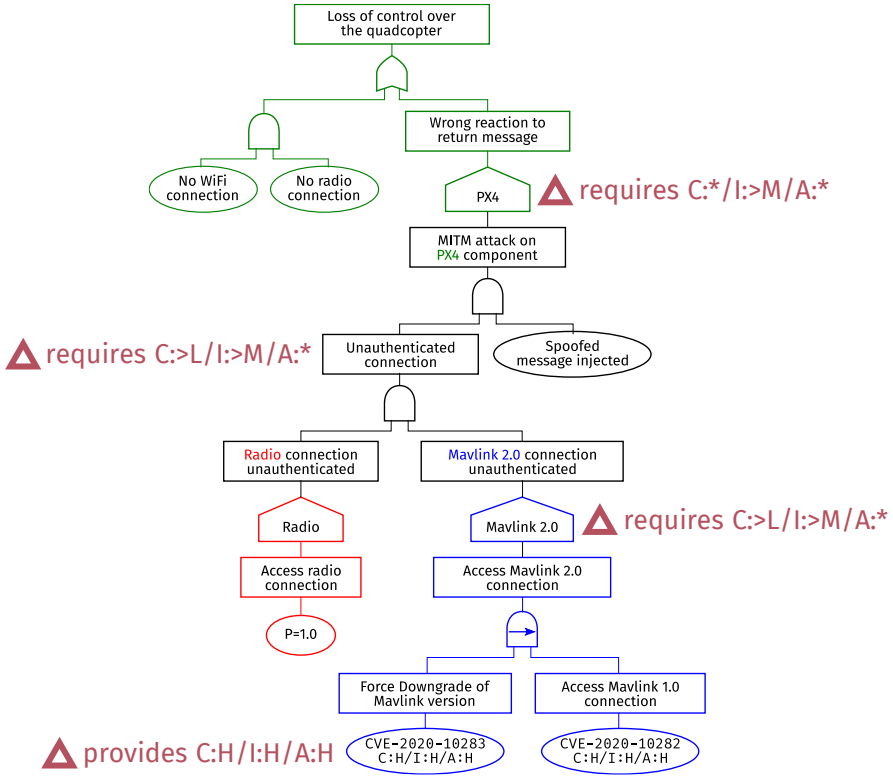
- Entity type: e.g. Library, Component, Channel…
- Name/CPE: Attacks reference CPE, Fault Trees reference dataflow elements by name
- CVSS impact metric: a fault needs a minimal impact on e.g. availability to trigger
- Possibly additional attack classification/taxonomy



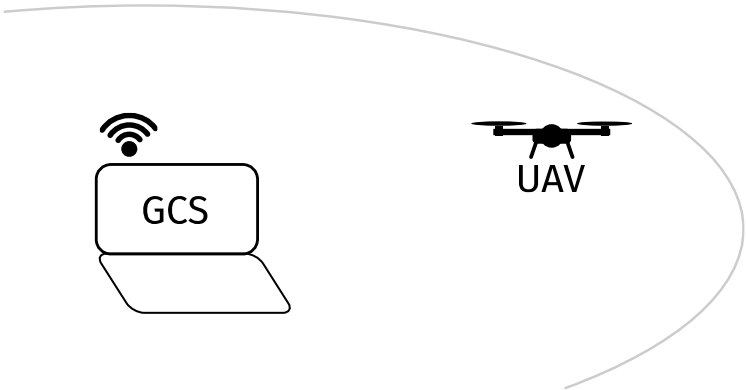
Compatibility Constraints

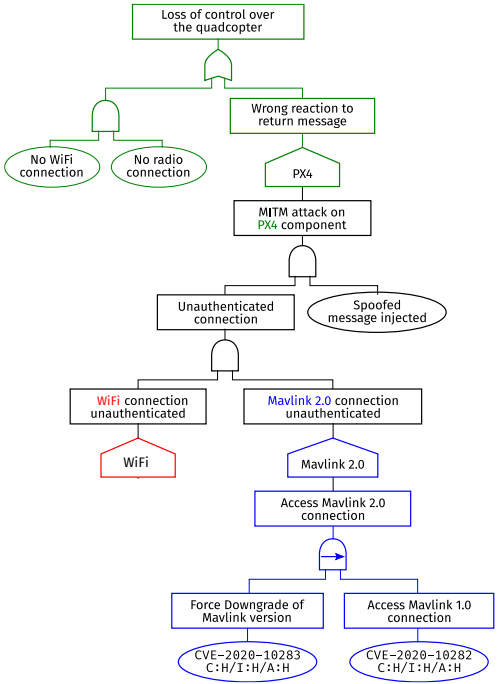
Rule application needs to be constrained

- Entity type: e.g. Library, Component, Channel…
- Name/CPE: Attacks reference CPE, Fault Trees reference dataflow elements by name
- CVSS impact metric: a fault needs a minimal impact on e.g. availability to trigger
- Possibly additional attack classification/taxonomy



Reconfiguration Example

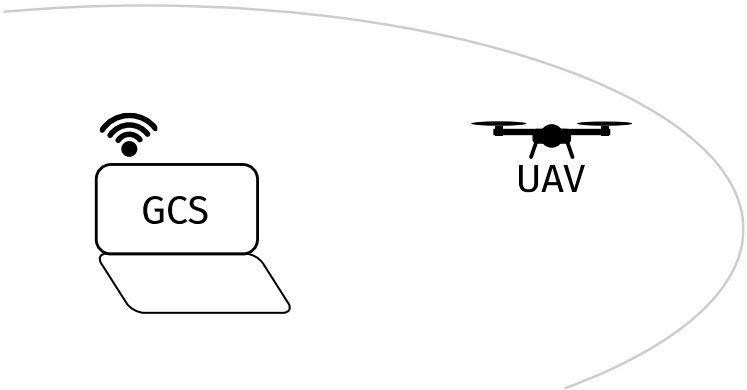


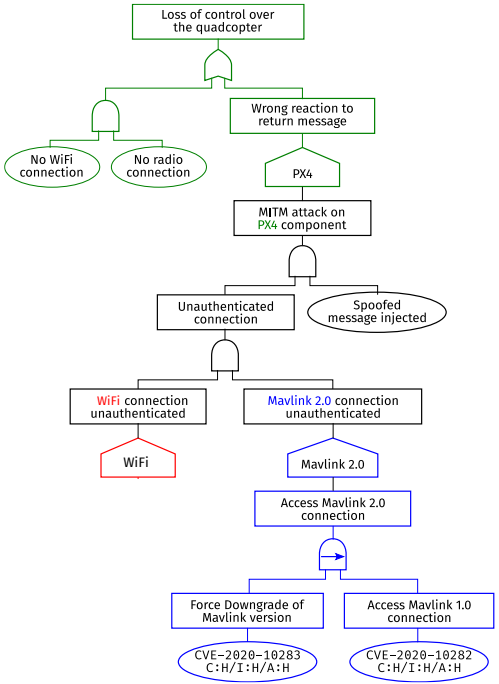


commands:Channel → {Mavlink2.0, WiFi}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

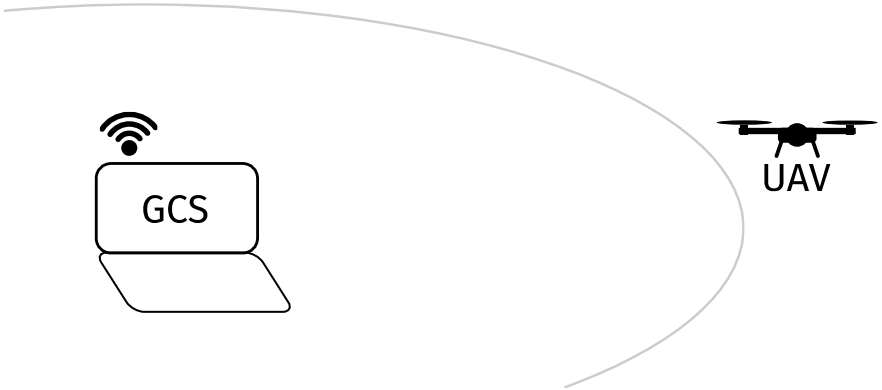


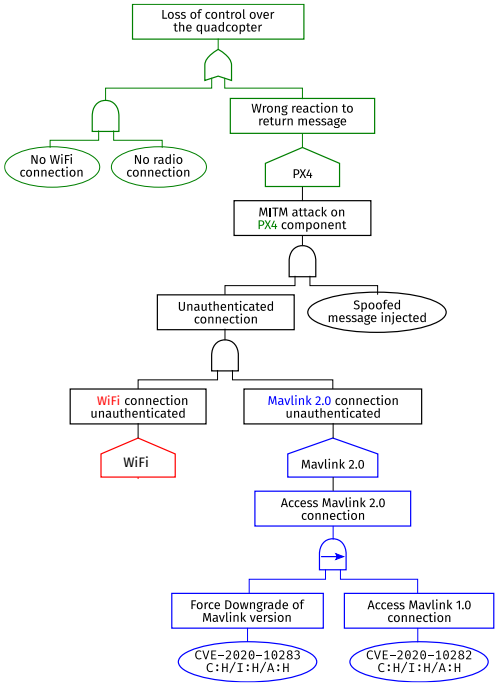


commands:Channel → {Mavlink2.0, WiFi}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

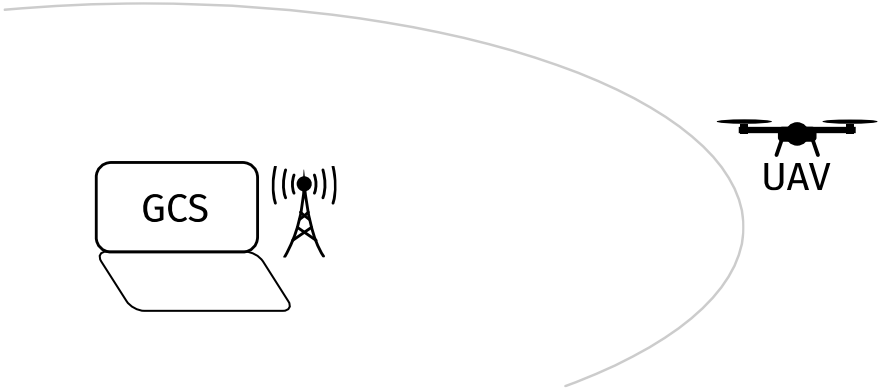


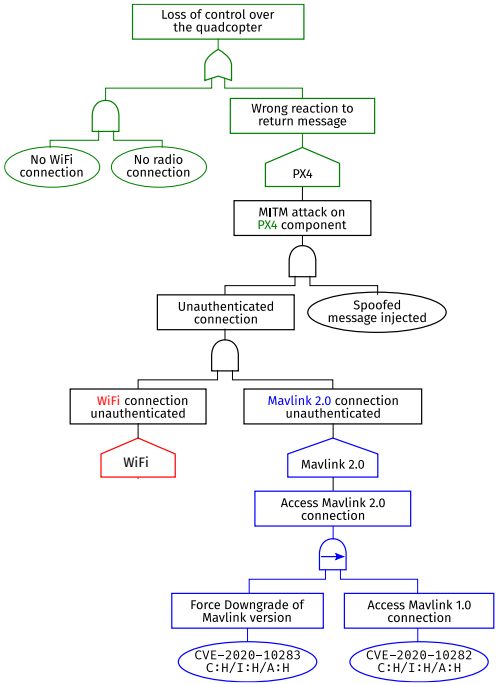


commands:Channel → {Mavlink2.0, WiFi}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

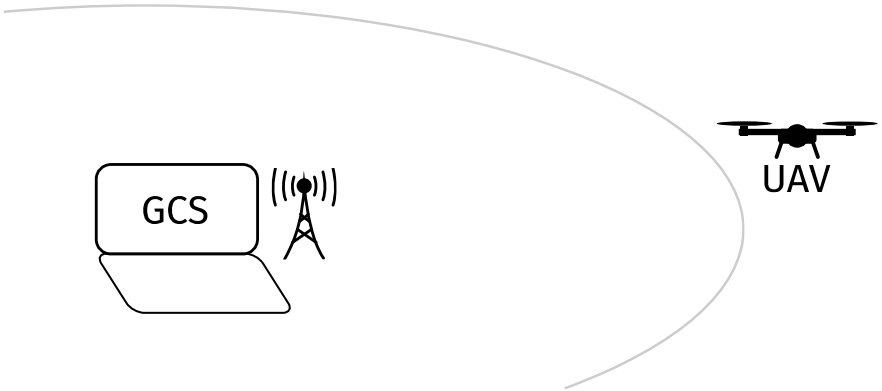


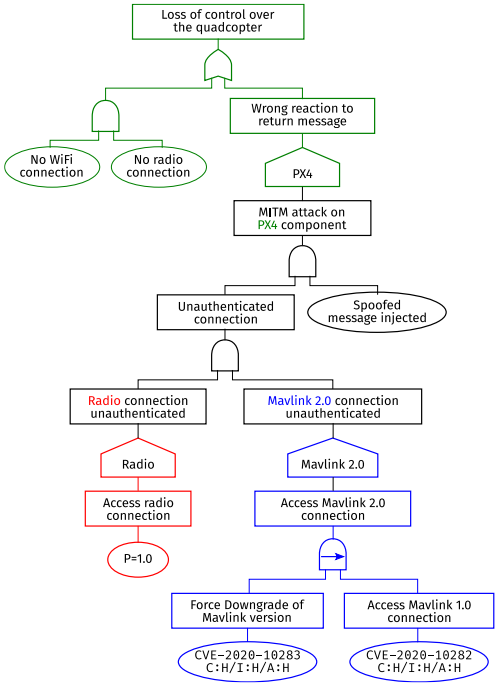


commands:Channel → {Mavlink2.0, WiFi}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

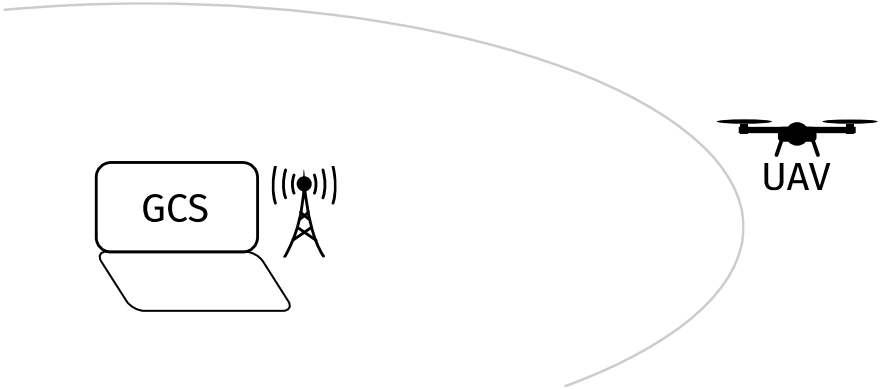


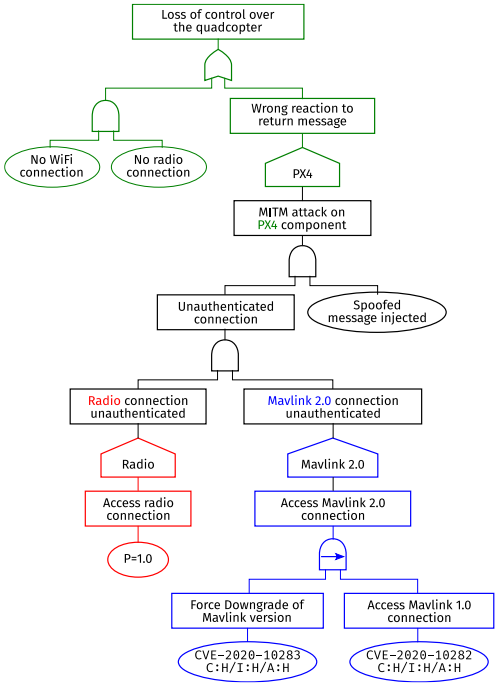


commands:Channel → {Mavlink2.0, Radio}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

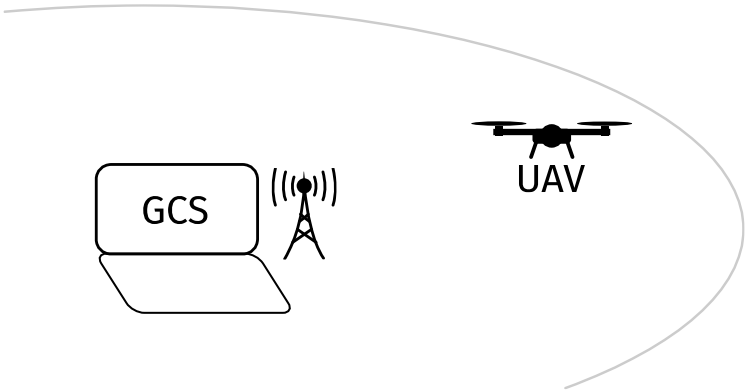


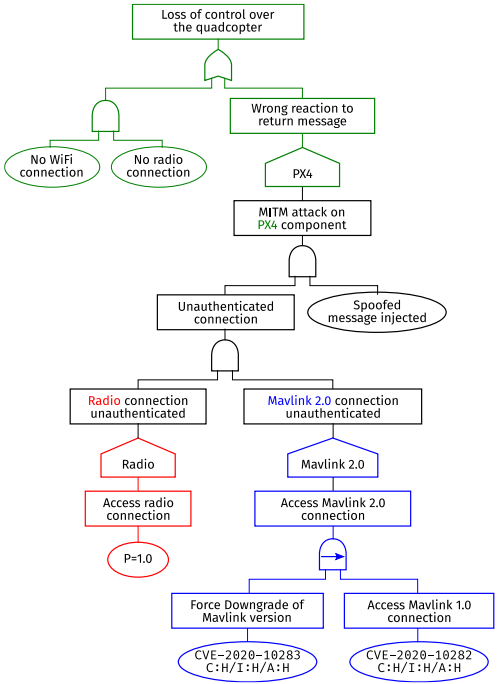


commands:Channel → {Mavlink2.0, Radio}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

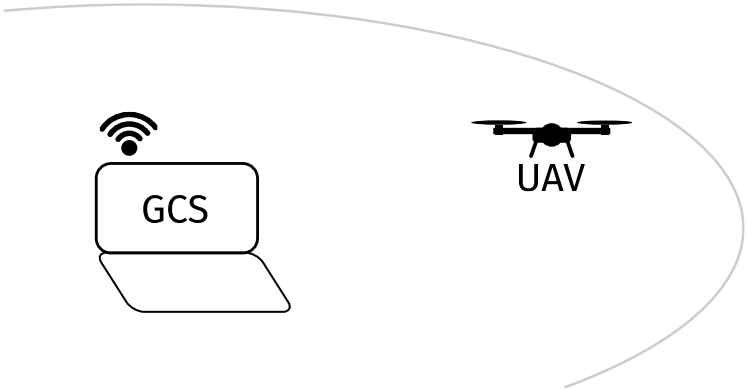


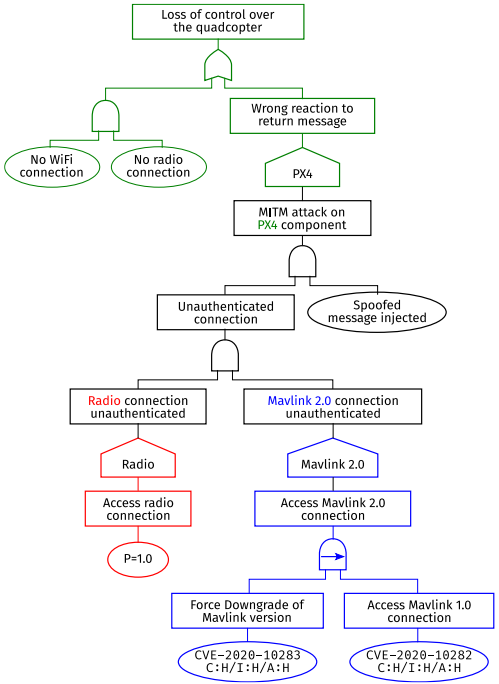


commands:Channel → {Mavlink2.0, Radio}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

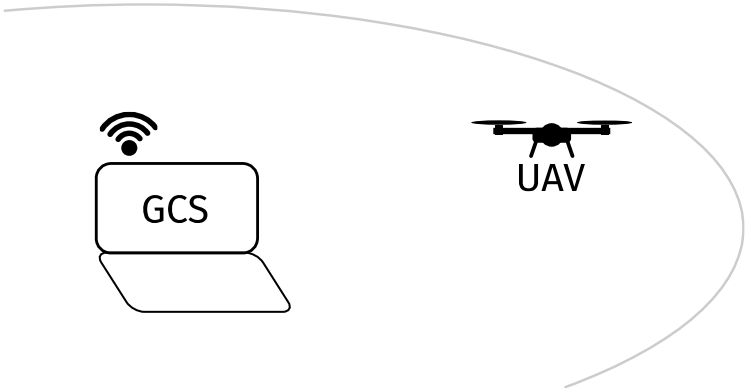


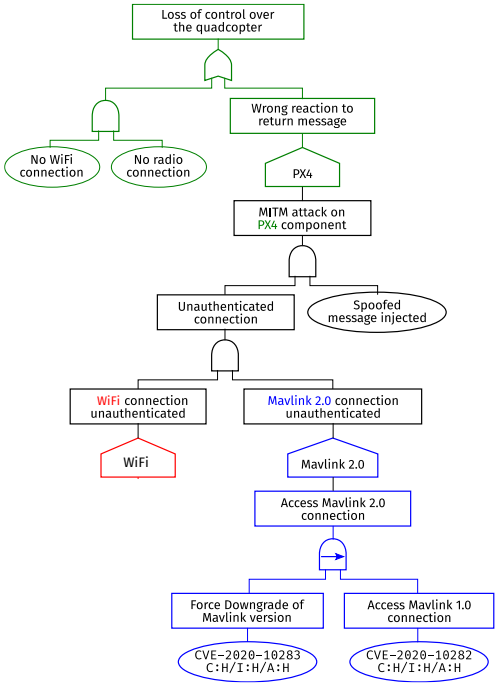


commands:Channel → {Mavlink2.0, Radio}

WiFi connected > Radio Connected > Return signal > WiFi connected

Reconfiguration Example

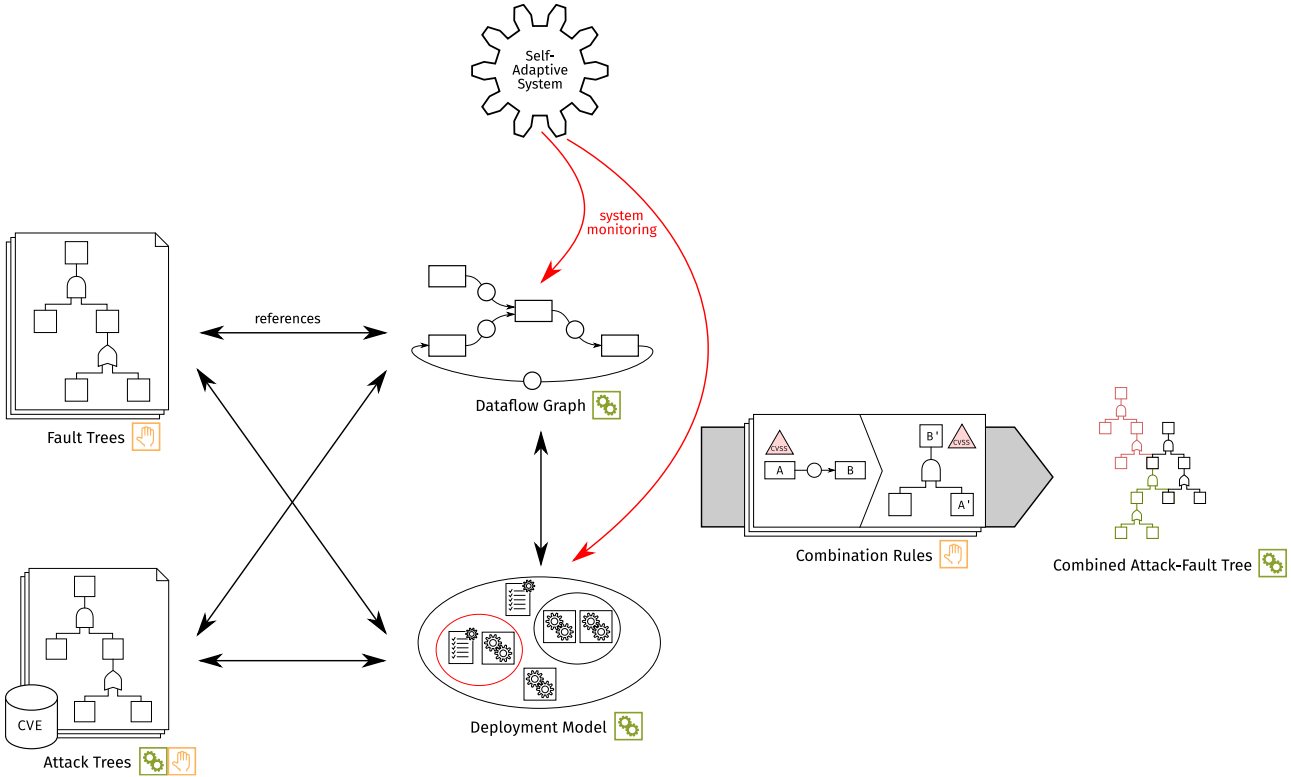




commands:Channel → {Mavlink2.0, WiFi}

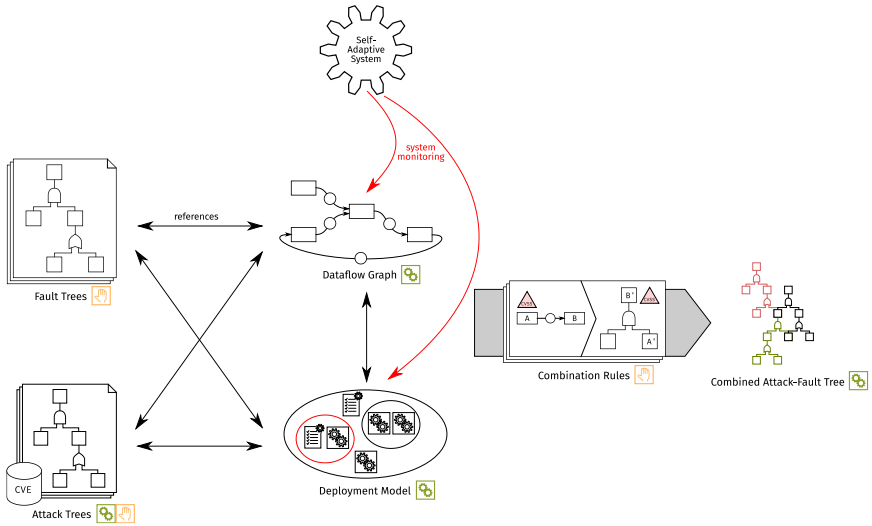
WiFi connected > Radio Connected > Return signal > WiFi connected

The SafeSec AFT Generation Toolchain (SAFT-GT)



Limitations & Future Work

- Behavioral models for channels and components
- Extend AFT properties (Time, Probability distribution)
- Realistic scenario and evaluation of generated AFTs
- Improvements to real-time analysis
- New approaches for generation of attack chains

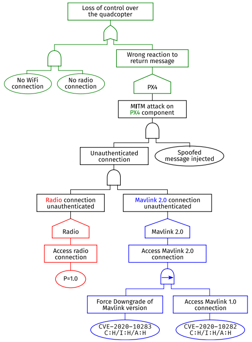


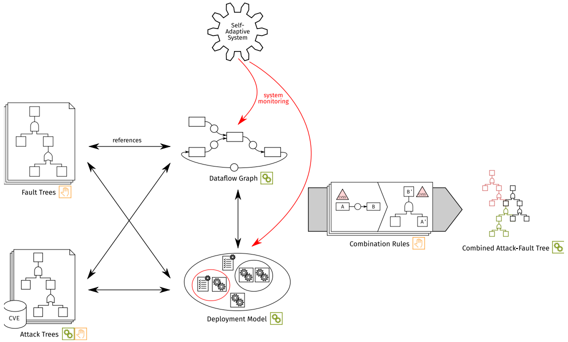
Conclusion

- AFT analysis integrates Safety and Security
- Generate AFTs to adapt to system reconfiguration

- Multiple Models help bridging different abstraction levels
- Rule based transformation to AFTs
- AFT regeneration for each system configuration

- Xtext implementation for each model
- Prototype of the AFT generation
- Generation of Attack Trees, Dataflow & Deployment models in progress






```
//1. Precondition: A component must be referenced
if(not self.oclIsTypeOf(dataflow::Component)){
    return patternList;
};
var comp:dataflow::Component:=self.oclAsType(dataflow::Component);

//2. Precondition: Component has one incoming channel
var inportList:=comp.elements->select(ele|ele.oclIsTypeOf(dataflow::InPort));
if(inportList->oclIsUndefined() or inportList->isEmpty()){
    return patternList;
};
var incomingChannelList:=sourceInDF.rootObjects()[dataflow::Model]
    ->allSubobjectsOfKind(dataflow::Connection)->flatten()
    ->select(con|con.port.oclIsTypeOf(dataflow::InPort) and
        inportList->includes(con.port)).channel;

//3. Precondition: Channel uses something related to an attack
var channelDefinitionList:=sourceInD.rootObjects()[deployment::Model]
    ->allSubobjectsOfKind(deployment::Element)->flatten()
    ->select(ele|ele.element.oclIsTypeOf(deployment::RefChannel) and incomingChannelList
        ->includes(ele.element.oclAsType(deployment::RefChannel).linkedDataflowChannel))
    ->select(ele|ele.checkForAffectedChannel())->asSequence();
```

universität innsbruck

UNIVERSITÄT LIECHTENSTEIN

universität ulm

Thank you for your attention!

Thomas Witte

Raffaela Groner

Alexander Raschke

Matthias Tichy

Irdin Pekaric

Michael Felderer

Sophie Hirn

Markus Frick