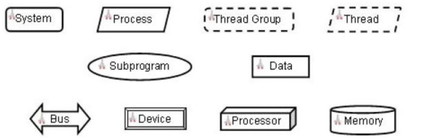
Modeling Security Aspects in Embedded Systems

Matt Eby, Jan Werner, Janos Mathe, Gabor Karsai, Sandeep Neema, Janos Sztipanovits, Yuan Xue
Institute for Software Integrated Systems, Vanderbilt University

Security Codesign Environment

Architectural Analysis and Design Language (AADL)

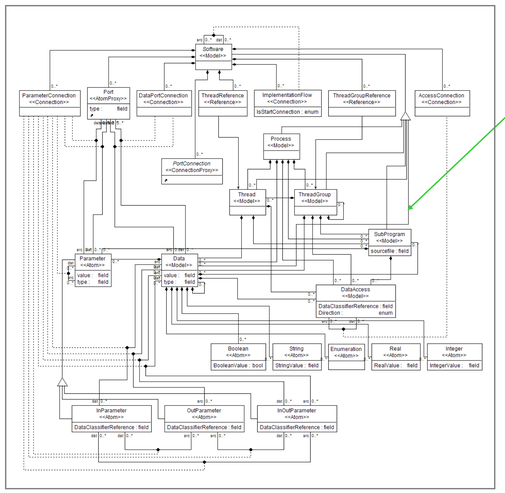
- Standard Specified by SAE Aerospace
- Developed to model embedded systems with challenging resource constraints

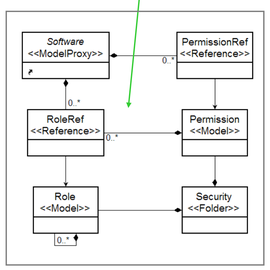


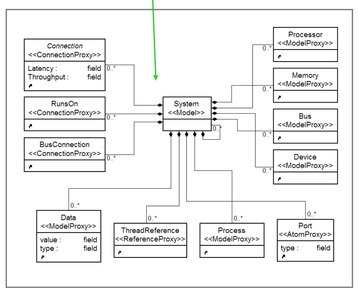
Role Based Access Control (RBAC) building blocks

- Objects – subject to access control
- Operations – execution of some functions on objects
- Permissions – approval to perform operation on RBAC protected object
- Role – job with assigned authority and responsibility
- User – human being, machine, network or agent requesting operation on objects

AADL with RBAC extensions have been implemented in the Generic Modeling Environment (GME). This allows system engineers to consider functionality and security at design time.

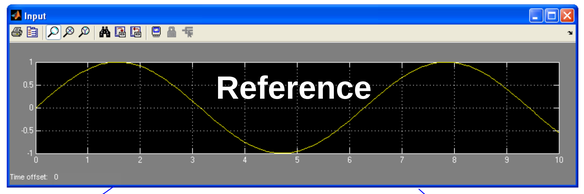




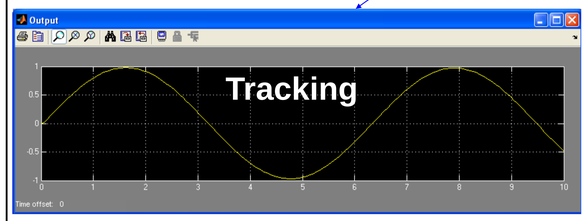


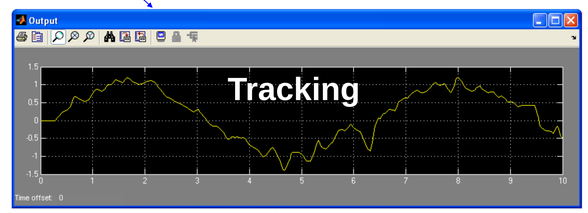
DC Motor Control Application

Correct Function of DC Motor Controller

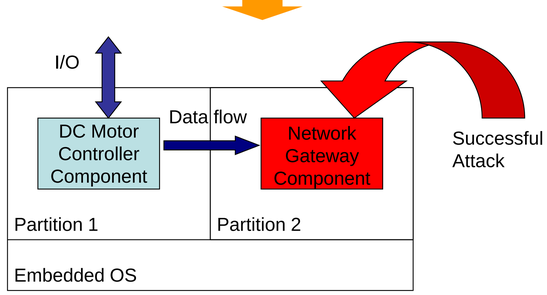


System Under Attack Loses Functionality





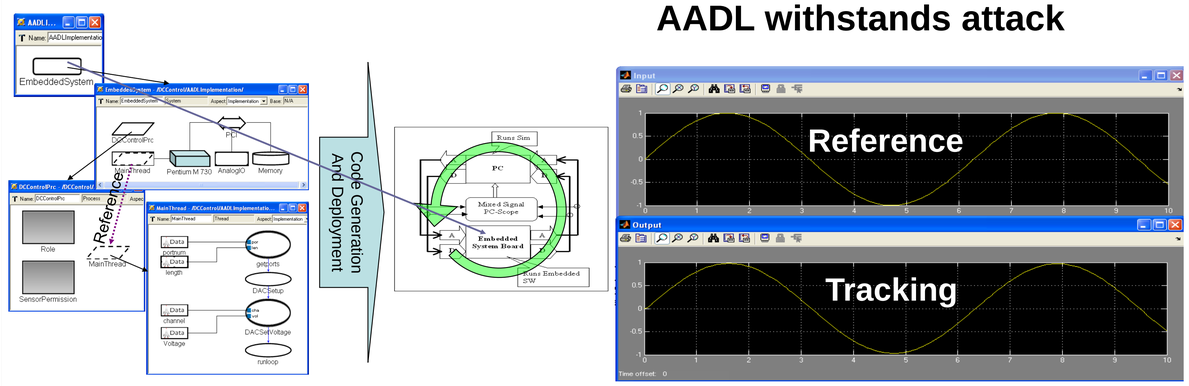
Components are placed in separate partitions. Successful attack on the network component is isolated in the partition.



The embedded controller shown above loses functionality when system resources are under attack. RBAC can protect system resources by only allowing privileged users access to the resources.

To prevent the waste of system resources by an unauthorized user the controller is modeled in AADL. RBAC is used to map permissions to users. The code generated from this model enforces the security policies and the system is resilient against attack.

System Designed RBAC in AADL withstands attack



Code Generators

DCControlPrc.h

```
#include "iolib.h"
#include "netserv.h"
#include "rbac.h"
pthread_t thread_CommThread;
int ret_CommThread;
pthread_t thread_MainThread;
int ret_MainThread;
void *CommThread(void);
void *MainThread(void);
```

DCControlPrc.c

```
#include "DCControlPrc.h"
void main(void){
requirepermission("SensorAccess");
retCommThread = pthread_create( &thread_CommThread,
NULL,CommThread, NULL);
retMainThread = pthread_create( &thread_MainThread,
NULL,MainThread, NULL);
pthread_join(thread_CommThread, NULL);
pthread_join(thread_MainThread, NULL);
}
void *CommThread(void){
startserver();
}
void *MainThread(void){
int length;
int portnum;
float VoltageLevel;
int Channel;
length=7;
portnum=0x350;
getports(length, portnum);
DACSetup();
Channel=1;
VoltageLevel=5;
DACSetVoltage(Channel, VoltageLevel);
runloop();
}
```

Code generators traverse the model and produce secure code that enforces the RBAC policies. The code generator makes use of the partitioning capabilities of the underlying platform.

makefile

```
CFLAGS = -lpthread
all:  Gateway DCControlPrc
Gateway: Gateway.c netclient.c netclient.c
DCControlPrc: DCControlPrc.c netserv.c iolib.c iolib.c iolib.c

install:
mkdir /DCControlPrc
mkdir /DCControlPrc/bin
cp DCControlPrc /DCControlPrc/bin
echo copy all required libraries to /DCControlPrc and use chroot()

.PHONY: all install
```