TCIPG

Real-Time Streaming Data Processing Engine for Embedded Systems

K. Tseng, Z. T. Kalbarczyk, and R. K. Iyer

Goals

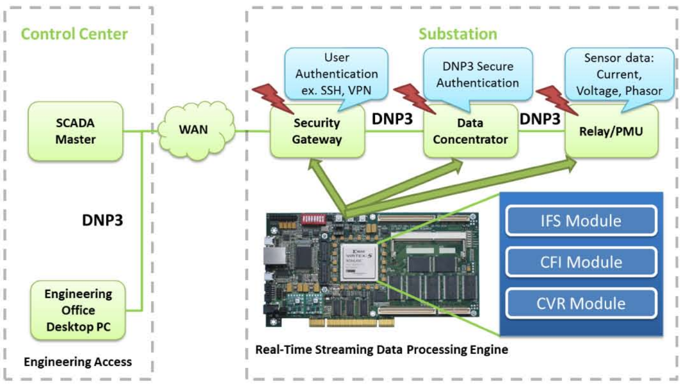
- Develop high-coverage and low-overhead techniques to achieve secure and reliable execution of applications that compute critical data, in spite of potential hardware and software vulnerabilities.
- Create a flexible, low-cost, and low-interference data processing engine for ensuring reliable and secure computing without incurring much resource and performance overhead.
- In particular, it should also be possible to insert the engine as a PCI Express card into substation devices, such as a Security Gateway, to protect the data stream against corruption due to accidental errors or malicious attacks.

Fundamental Questions/Challenges

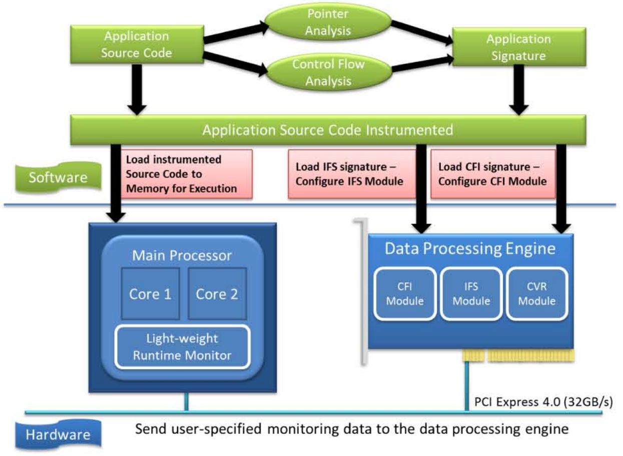
- How to protect critical power grid data from malicious tampering and transient errors with high coverage.
- How to prevent attackers from using different entry paths (outsiders, normal users, or insiders).
- How to design the Data Processing Engine so that it can efficiently collect monitoring data coming from the main processor of the substation devices with low interference and low overhead.
- How to achieve low-cost, high-performance, flexible, and scalable security and reliability checking.

Research Plan

- **Information Flow Signature (IFS):** Perform analysis of the high-level source code of a program to derive the IFS (the set of data objects that one instruction is allowed to access) to ensure that runtime modifications of the data object follow the language-level semantics of the application.
- **Control Flow Integrity (CFI):** Develop a runtime check to ensure that the execution of the program follows the Control-Flow Graph (CFG) extracted from the high-level source code. In particular, only indirect transitions, such as indirect jump or function return, need to be checked.
- **Critical Value Re-computation (CVR):** Statically analyze the application source code to extract the backward slices for each piece of critical data, which will be recomputed in runtime by a computation path different from the original one to prevent soft errors, such as transient errors.


Control Center
SCADA Master
WAN
DNP3
Engineering Office Desktop PC
Engineering Access
Substation
User Authentication ex. SSH, VPN
DNP3 Secure Authentication
Sensor data: Current, Voltage, Phasor
Security Gateway
DNP3
Data Concentrator
DNP3
Relay/PMU
IFS Module
CFI Module
CVR Module
Real-Time Streaming Data Processing Engine

Design and Integration of Real-Time Data Processing Engine


Application Source Code
Pointer Analysis
Control Flow Analysis
Application Signature
Application Source Code Instrumented
Software
Load instrumented Source Code to Memory for Execution
Load IFS signature – Configure IFS Module
Load CFI signature – Configure CFI Module
Main Processor
Core 1
Core 2
Light-weight Runtime Monitor
Data Processing Engine
CFI Module
IFS Module
CVR Module
PCI Express 4.0 (32GB/s)
Hardware
Send user-specified monitoring data to the data processing engine

Research Results

- A simplified version of the IFS module has been synthesized on Altera Startix II FPGA running SSH, WuFTP, and NullHTTP on top of Linux to demonstrate the effectiveness of IFS with low overhead (3–4%) and high coverage.

Attack	Target	Difficulty of launching attacks	Detectability
Control-data attacks	Return address	Low	Yes
	Function pointer	Low	Yes
	Old base pointer	Low	Yes
	Longjmp buffer	Low	Yes
Non-control-data attacks	Information signature	High	No
	Configuration data	Low	Yes
	User input	Low	Yes
	User identity data	Low	Yes
	Decision-making data	Low	Yes
Code injection attacks	Code segment	Moderate	Yes (for certain class)
Register corruption attacks	Register	High	No
Malicious 3rd-party library	Shared library (control flow reserved)	Moderate	Yes
	Shared library (control flow changed)	Moderate	No (detectable by CFI)
SETUID attacks	SETUID programs	Moderate	No

Broader Impact

- The fact that the Real-Time Streaming Data Processing Engine works as a PCI-E card and only requires slight modifications to the main processor makes it very easy for the engine to protect different types of power grid systems.

Interaction with Other Projects

- We utilize the methods and tools developed by the activity “Testbed-Driven Assessment” to set up our experimental environment.

Future Efforts

- Implement the complete version of the Real-Time Streaming Data Processing Engine and apply it on power grid substation devices, such as security gateways or data concentrators.
- Evaluate the detection coverage of the engine against the latest real attacks.