DIAT:
Data Integrity Attestation for Resilient Collaboration of Autonomous Systems

Tigist Abera, Raad Bahmani, Ferdinand Brasser, 
Ahmad Ibrahim, 
Ahmad-Reza Sadeghi, and Matthias Schunter

Technische Universität Darmstadt, Germany and 
Intel Labs, Portland, OR, U.S.A.
Motivation
Remote attestation checks trustworthiness of a remote (embedded) device
Remote Attestation

Interactive Protocol

Prover

Verifier

H = hash()

R = mac(N, H)

N

R

Random Nonce

Attestation Response

Is the prover in the correct state?
Key Limitation:

Static attestation schemes do not address runtime attacks
Problem Space of Runtime Attacks

Control-Flow Attack
[Shacham, ACM CCS 2007]
[Schuster et al., IEEE S&P 2015]

Non-Control-Data Attack
[Chen et al., USENIX Sec. 2005]
[Carlini et al., USENIX Sec. 2015]

Basic Block
switch(opmode)
case recovery: C
case op1: D
case op2: E,F

Adversary
DEP

corrupt code
corrupt data
pointer/variable
inject malicious code
Memory write
Program flow
Control-flow attestation aims at the detection of runtime attacks
Control-Flow Attestation

Cumulative Hash Value: $H_i = H(H_{i-1}, N)$

- $H_{i-1}$ -- previous hash result
- $N$ -- instruction block (node) just executed

$H_1 = H(0, A)$
$H_2 = H(H_1, B)$
$H_3 = H(H_2, C)$
$H_4 = H(H_2, D)$
$H_5 = H(H_2, E)$
$H_6 = H(H_5, F)$
Problems
Control-Flow Attestation

- High overhead on the verifier
- Program complexity leads to a large number of valid hashes
- Only applicable to small programs
Control-flow attestation for autonomous systems
High Level Idea

**Modularization**
Software is divided into smaller isolated modules

**Data-flow attestation**
Attestation is executed when data is exchanged

**Exec path representation**
Execution path is represented as a multiset of edges
Modularization
Software is divided into smaller isolated modules

Modular software
can be decomposed into simple interacting modules

Data-flow monitoring
Software modules interact through a well-defined communication channels

Isolation Architecture
Software modules are securely isolated for each other

Data-flow attestation
Attestation is executed when data is exchanged

Exec path representation
Execution path is represented as a multiset of edges
Module: Data-Flow Monitoring

**Modularization**
Software is divided into smaller isolated modules.

**Data-flow attestation**
Attestation is executed when data is exchanged.

**Exec path representation**
Execution path is represented as a multiset of edges.
Control-Flow Monitoring

Modularization
Software is divided into smaller isolated modules

Data-flow attestation
Attestation is executed when data is exchanged

Exec path representation
Execution path is represented as a multiset of edges
High Level Idea

DFMonitor
M₃ → M₅
M₅ → M₆
M₆ → M₃

CFMonitor
M₃ M₅ M₆

17
Implementation
Autonomous Drones

Pixhawk: open-hardware project autopilot hardware

PX4: open source flight control software for drones
Objective
Observes data flow between software modules and identify critical ones

Realization
Extending Middleware to enable data-flow monitoring functionalities

Functionalities:
- Extending MAVLink message format to include attestation requests/response
- Extending uORB to record message subscription and data generation
- Flushing uORB data buffers before when sensitive data is requested
Extending MAVLink message format

Flushing uORB data buffers

Observing data flow between modules
Objective
Observes execution of critical modules and records their control flow

Realization
Instrumenting software modules with instructions that allow recording its control flow

Functionalities:
- Logic for recording the control flow events of critical modules
- Instrumentation instruction which call the logic at every control-flow event
**CFMonitor**

- **Critical Modules**:
  - ID: 1
  - MSHV: mshv1
  - Path: \(<e1, 1>, <e3, 1>, <e4, 1>\)
  - ID: 3
  - MSHV: mshv3
  - Path: \(<e2, 1>, <e4, 1>, <e5, 1>\)
  - ID: 5
  - MSHV: mshv5
  - Path: \(<e1, 2>, <e2, 1>, <e4, 1>\)

- **Process Flow**:
  - Start
  - Input CF-event
  - Module critical?
  - Update MSH-Value
  - Update Path
Integration into PX4
Concept

Flight Stack

Module 1

Middleware

MAVLink
Request
Response

DFMonitor
Filter
uORB

CFMonitor
MSH
Quoter

NuttX
Evaluation
### GPS Coordinates

<table>
<thead>
<tr>
<th>MODULE</th>
<th>CFG SIZE</th>
<th>EXECUTION PATH</th>
<th>ATTESTATION TIME</th>
<th>VERIFICATION TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>2922</td>
<td>22249</td>
<td>835</td>
<td>849</td>
</tr>
<tr>
<td>GYROSCOPE</td>
<td>912</td>
<td>20004</td>
<td>748</td>
<td>760</td>
</tr>
<tr>
<td>E-COMPASS</td>
<td>1468</td>
<td>18907</td>
<td>716</td>
<td>718</td>
</tr>
<tr>
<td>IMU SENSOR</td>
<td>1905</td>
<td>158671</td>
<td>6341</td>
<td>6357</td>
</tr>
<tr>
<td>PRESSURE SENSOR</td>
<td>1051</td>
<td>1150</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>FMU</td>
<td>1828</td>
<td>38132</td>
<td>1510</td>
<td>1511</td>
</tr>
<tr>
<td>PX4IO</td>
<td>3661</td>
<td>12723</td>
<td>484</td>
<td>489</td>
</tr>
<tr>
<td>LED DRIVER</td>
<td>532</td>
<td>32</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>STM32 ADC</td>
<td>251</td>
<td>21274</td>
<td>805</td>
<td>808</td>
</tr>
<tr>
<td>COMMANDER</td>
<td>7852</td>
<td>9418</td>
<td>354</td>
<td>365</td>
</tr>
<tr>
<td>LOAD MONITOR</td>
<td>135</td>
<td>8</td>
<td>0,3</td>
<td>0,4</td>
</tr>
<tr>
<td>SENSORS</td>
<td>2032</td>
<td>40410</td>
<td>1618</td>
<td>1623</td>
</tr>
<tr>
<td>SYSTEMLIB</td>
<td>2555</td>
<td>662142</td>
<td>26341</td>
<td>26365</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>27014</strong></td>
<td><strong>1005120</strong></td>
<td><strong>39799,3</strong></td>
<td><strong>39892,4</strong></td>
</tr>
</tbody>
</table>

GPS coordinates involves **1 of 13** executing modules

Modularity entails an improvement of **95%** on runtime
## Different Data Types

<table>
<thead>
<tr>
<th>Data</th>
<th>cmd_state</th>
<th>battery_status</th>
<th>sensor_acel</th>
<th>sensor_gyro</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical Modules</td>
<td>12</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Executed Modules</td>
<td>12</td>
<td>13</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Percentage</td>
<td>100%</td>
<td>92%</td>
<td>28%</td>
<td>25%</td>
</tr>
<tr>
<td><strong>∑ of CFGs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical Modules</td>
<td>197823</td>
<td>46860778</td>
<td>194</td>
<td>250</td>
</tr>
<tr>
<td>Executed Modules</td>
<td>197823</td>
<td>46862156</td>
<td>1590</td>
<td>1328</td>
</tr>
<tr>
<td>Percentage</td>
<td>100%</td>
<td>99%</td>
<td>12%</td>
<td>18%</td>
</tr>
<tr>
<td><strong>∑ of Executed Paths</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical Modules</td>
<td>26572</td>
<td>26572</td>
<td>3373</td>
<td>2817</td>
</tr>
<tr>
<td>Executed Modules</td>
<td>26572</td>
<td>27104</td>
<td>13622</td>
<td>13873</td>
</tr>
<tr>
<td>Percentage</td>
<td>100%</td>
<td>98%</td>
<td>24%</td>
<td>20%</td>
</tr>
</tbody>
</table>
Hybrid Serial Parallel Collaboration (no security) Devices recursively request GPS coordinates Authentication Exchanged data is authenticated with ECDSA DIAT The control flow of critical modules is attested
Logarithmic in hybrid collaboration

4x authentication based on ECDSA
Security
Security Considerations

DFMonitor:
- All critical modules will be detected and attested

CFMonitor:
- Adding edges *not* in CFG will be detected
- Adding edges in CFG to execution path requires security policy
- Reordering edges in the execution path *cannot* be detected
Conclusion
Conclusion

Static attestation cannot detect runtime attacks

Control-flow attestation (CFA) is too complex

DIAT allows CFA in the autonomous settings. However, this requires

• Modular software design with clear communication

• Strong isolation between software modules