A Testing Framework Architecture Concept for Automotive Intrusion Detection Systems

30.05.2017 - Automotive 2017
Agenda

- Introduction
- Intrusion detection systems scope
- Problem statement
- Our approach
- Conceptional architecture
- Conclusion
- Outlook
Introduction
Speakers

Corbett Christopher

- Phd Student at the University of Ulm
- Automotive security engineer at Audi AG
- ~ 15 years of experience in the automotive industry
- christopher.corbett@audi.de

Basic Tobias

- Completed double masters degree on IT security and computer science at the university of Darmstadt
- Automotive specialist security and privacy at Continental AG
- fi59eged@rbg.informatik.tu-darmstadt.de
Intrusion Detection Systems in In-Vehicle Networks

Scope

Mainly focusing on Network Intrusion Detection Systems (NIDS) and anomaly detection
Evaluation during development phase is very difficult as real data and attacks do not exist!
Intrusion Detection Systems in In-Vehicle Networks

Automotive protocols

- There is a large variety of protocols to consider!
- Some protocols are automotive only!
Problem Statement

- In-vehicle network traffic not publicly available for use
- Automotive network topologies differ from OEM to OEM
- Sharing of information, especially during development phase, is prohibited and often part of intellectual property (IP)
- New technologies (e.g. Ethernet, CAN-FD) and protocols (e.g. SOME-IP) can’t easily be evaluated
- Malicious traffic barely exists
- Complexity of in-vehicle attacks is different to existing attacks

Dependencies for an NIDS evaluation are not fulfilled!
Our approach
Preparation

- Malicious traffic
- Valid traffic
- Evaluation metrics

NIDS Evaluation Requirements

- Automotive attack scenarios
- Applicable in real vehicles

Attack Requirements

- Support for required protocols
- Support for required technologies
- Use case coverage
- Realism

Network Traffic Requirements

- Available tools
- Supported platforms
- Available libraries
- User perspective

Miscellaneous
## Our approach

### Evaluation of existing Tools

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tomahawk</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit-Twist</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hping2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hping3</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nemesis</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ostinato</td>
<td>✔</td>
<td>(✔)</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(✔)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>packETH</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yersinia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td>(✔)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>netsniff-ng</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pktgen</td>
<td>✔</td>
<td>(✔)</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Existing tools don’t cover the necessary requirements!
Our approach

Practical example

- Import some libraries
- Create can socket
- Create ethernet socket
- Read from network interface CAN0
  ...
  Create ethernet frame and upper layers
  Set IP address, VLAN Tag, etc.
  create payload and do some processing
  ...
  Add payload to frame
- Send frame via socket
- Close socket

Parameterize topology information
Our approach
Practical example

Scenario

Network/Topology parameters
Functional parameters
System parameters
Custom settings
Architecture Concept
Our approach

Summary

- Architecture only based on open source components
- Support of several network interfaces (Ethernet, CAN, WiFi, USB, ...)
- Separation of use case logic, operating system dependencies and network stacks
- Create realistic automotive traffic for Network Intrusion Detection Systems (NIDS)
- Providing If-Than-Else Functionality
- Separation of network topology information and use case description
- Encapsulate logic in function blocks
- Enable simple fuzzing functionality
- Sharing implementations, setups and (if possible) datasets with the community
Conclusion

Feasible:
- If-than-else functionality
- Scene description
- Capsulating functionality
- Using open source software only
- Concept architecture implementation

Challenging:
- Parallelism
- Message/Interface priorities
- Timing
- Library support of protocols
Outlook

First prototype called (anxiety) using python3
Publication of the source code on a collaboration plattform (pending)

Final master thesis available soon on the university of Darmstadt website containing:
- Detailed descriptions
- Performance measurement

Publication of a follow up document about the implementation and evaluation of the prototype (currently working on it)
Thank you