Operational Design for Advanced Persistent Threats

Tithnara Nicolas SUN
Ciprian TEODOROV

Luka LE ROUX

19/10/2020
Advanced Persistent Threat

• Specific targets and clearly defined goals

• Highly organized and well-resourced attackers

• Long-term campaigns with repeated attempts

• Stealth and evasion tactics

(NIST, 2011)
# APT – Solutions

(Brewer et al., 2014)

<table>
<thead>
<tr>
<th>Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reconnaissance &amp; weaponization</td>
</tr>
<tr>
<td>Delivery</td>
</tr>
<tr>
<td>Initial intrusion</td>
</tr>
<tr>
<td>Command &amp; control</td>
</tr>
<tr>
<td>Lateral movement</td>
</tr>
<tr>
<td>Data exfiltration</td>
</tr>
</tbody>
</table>
APT – Limits

Reconnaissance  ?  Weaponization

Strategy
Operational Design

(Graves et al., 2013)
Operational Design

Mission statement

Objective identification

Frame the problem

Target identification

Frame the current operational environment

Frame the desired operational environment

Operational approach
Pimca Framework

• Systems modeling language
  • High-level of abstraction
  • Graphical
  • Geared toward security

(Sun et al., 2020)
Pimca Framework

• Dynamic extension requirement
  • System behavior framing
  • Desired environment framing
  • Problem framing
A **behavioral model** is defined as:

\[ M = < V, A, S > \]

- \( V \) is a set of variables
  - \( val_V \) is the set of possible valuations over \( V \)
- \( A \) is a set of guarded-commands
- \( S \) is a set of synchronisation channels
A **guarded-command** is defined as:

\[
G_c = < u, s, g, c >
\]

- \( u : \mathbb{B} \), denotes if \( G_c \) is urgent
- \( s : S \cup \{none\} \), is a synchronisation channel (or absence of)
- \( g : \text{val}_V \rightarrow \mathbb{B} \), is a boolean expression of the model variables
- \( c : \text{val}_V \rightarrow \text{val}_V \), is a sequence of statements

**GC_name:**

```plaintext
urgent ?
(channel ( ? ! ! )) ?
[guard] ? /
(command ;) *
```
Execution rules:

• A guarded-command can only be executed if its guard is true on the current valuation.

• Only one guarded-command can be executed at a time.

• If a guarded-command uses a synchronisation channel, it must be executed sequentially in a single step alongside a synced guarded-command in the following order: (emission, reception).

• If any urgent guarded-command can be executed on the current valuation, the next execution step must involve an urgent guarded-command.
Case study

Water pumping station

Frame the current operational environment
Case study

Water pumping station

Water tank

Role: to update the `waterLevel` variable

- `flowIn`
- `flowOut`
- `refreshSensor`
- `overflow`
- `underflow`
Case study

Water pumping station

PLC

Role: to control the water flow through actuators and sensors

- update
- regular
- highThreshold
- lowThreshold
- valveOn
- valveOff
- pumpOn
- pumpOff

Frame the current operational environment
### Water pumping station

<table>
<thead>
<tr>
<th>WaterTank</th>
<th>PLC</th>
<th>InflowValve</th>
<th>ManualValve</th>
<th>Pump</th>
<th>Sensor</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>flowIn</td>
<td>update</td>
<td>flowOut</td>
<td>flowIn</td>
<td>flowIn</td>
<td>update</td>
<td>input</td>
</tr>
<tr>
<td>flowOut</td>
<td>regular</td>
<td>open</td>
<td>flowOut</td>
<td>open</td>
<td>refreshPLC</td>
<td></td>
</tr>
<tr>
<td>refreshSens</td>
<td>highThres</td>
<td>close</td>
<td>open</td>
<td>close</td>
<td></td>
<td></td>
</tr>
<tr>
<td>overflow</td>
<td>lowThres</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>underflow</td>
<td>valveOn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>valveOff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pumpOn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pumpOff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Water pumping station

Desired environment:
• Overflow the water tank
• Remain undetected

Expressed using LTL:
\((\Diamond \text{overflow}) \land (\Box ! \text{alert})\)
Case study

Water pumping station

Leverage capabilities:

• force the inflow valve open
• block the pump
• close the manual valve
• disable the sensor
• jam the network

<table>
<thead>
<tr>
<th>Inflow Valve</th>
<th>Pump</th>
<th>Sensor</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>force Open</td>
<td>block</td>
<td>disable</td>
<td>jam</td>
</tr>
<tr>
<td>close*</td>
<td>open*</td>
<td>refresh PLC*</td>
<td>send*</td>
</tr>
</tbody>
</table>
Case study

Water pumping station

Model-checking using OBP2:

<table>
<thead>
<tr>
<th>Force (open) inflow valve</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>Close manual valve</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block pump</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jam network</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disable sensor</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Sub-objective 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-objective 2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

TABLE 2: Model-checking of the water pumping station (O: success, X: failure)

Operational approach: disabling the sensor is the simplest path to achieving the mission
Conclusion

Modeling the APT strategy planning
- Adapted from Operational Design
- Pimca framework
- Model-checking

Future works
- Methodology refining, user study
- Problem framing formalization
Bibliography


• Class diagram
Execution rules

\[ \forall \langle \alpha, g, c \rangle \in A, \forall \rho_1, \rho_2 \in \text{val}_V \\
\text{single}_u : \\
u \wedge g(\rho_1) \wedge c(\rho_1) = \rho_2 \\
\langle \| , \rho_1 \rangle \rightarrow \rho_2 \]

\[ \forall \langle \alpha, g, c \rangle \in A, \forall \rho_1, \rho_2 \in \text{val}_V \\
\text{single} : \\
\neg \text{hasUrgent}_A(\rho_1) \wedge \neg u \wedge g(\rho_1) \wedge c(\rho_1) = \rho_2 \\
\langle \| , \rho_1 \rangle \rightarrow \rho_2 \]

\[ \forall \langle \alpha, (\text{out}, id), g, c \rangle, \langle \alpha, (\text{in}, id), g, c \rangle \in A, \forall \rho_1, \rho_2 \in \text{val}_V \\
\text{sync}_u : \\
(u_1 \lor u_2) \wedge g_1(\rho_1) \wedge g_2(\rho_1) \wedge c_2(c_1(\rho_1)) = \rho_2 \\
\langle \| , \rho_1 \rangle \rightarrow \rho_2 \]

\[ \forall \langle \alpha, (\text{out}, id), g, c \rangle, \langle \alpha, (\text{in}, id), g, c \rangle \in A, \forall \rho_1, \rho_2 \in \text{val}_V \\
\text{sync} : \\
\neg \text{hasUrgent}_A(\rho_1) \wedge \neg (u_1 \lor u_2) \wedge g_1(\rho_1) \wedge g_2(\rho_1) \wedge c_2(c_1(\rho_1)) = \rho_2 \\
\langle \| , \rho_1 \rangle \rightarrow \rho_2 \]