Models of reliability of fault-tolerant software under cyber-attacks

Peter Popov,
Centre for Software Reliability

23 March 2017
Urbana-Champaign
Software Reliability in Adverse Environment

- A significant effort on “co-engineering” for safety – reliability-performance has been spent
  - People talk about trade-offs, interdependencies between multiple non-functional properties, but models which make the dependence explicit do not seem to exist
  - Safety standards (e.g. ISO 61508, the main safety standard) recommend that “all risk including from malicious activities” be taken into account when engineering system safety.
  - The industrial practice is not ideal – different “silos” make it difficult to co-engineer for safety and security since different departments deal with different concerns.
Motivation

- In this work I make an attempt to model the impact of successful cyber-attacks on software reliability \textit{explicitly}.
- The work targets primarily \textit{industrial control system},
  - These must work reliably, availability is a paramount concern
  - privacy and confidentiality \textbf{are typically not} a primary concern
    - There may be exceptions – smart metering infrastructure is an example.
- Having reviewed a large number of reports of attacks on industrial control system convinced me that the model put forward is \textit{plausible}.
  - Full validation of the assumptions is yet to be done.
System model

- 1-out-of-2 “on demand” software
- Popular in safety critical applications, e.g.
  - Protection systems
  - Automotive industry (ASIL-D according to ISO 26262)
  - Etc.
System model (2)

- Channels can be subjected to cyber attacks (i.e. malicious demands)
- The view taken by many is: “once a malicious demand succeeds, the game is over: the adversary can do whatever they please”
  - The consequences of successful malicious demands are not modelled in detail and “the worst” consequences are assumed.
- In this work I take a different view:
  - Successful demands merely increase the probability of failure on “normal” (i.e. non-malicious) demands.
    - Immediate failure after a successful attack is merely a special case of extreme reliability decay: the probability of failure on demand increases to 1 (deterministic failure).
**System model (3)**

- Channels are “diverse” (using software design diversity)
  - Design diversity has been studied very extensively at City
  - Diversity *does not guarantee failure independence*
- Demands are *independently selected from the demand space* and processed by each channel independently

- The system fails when both channels fail simultaneously on the same demand
  - The probability of system failure on demand $X$ ($pf_d$) is:

\[
P(\pi_A, \pi_B \text{ fail on } X) = pf_d_A pf_d_B + \text{cov}_Q(\omega_A(X, \pi_A), \omega_B(X, \pi_B))
\]
Probability of failure on demand
System under-attack

- Malicious demands (MD), \{\mu_1, \mu_2, \ldots, \mu_n\} can be applied to each of the channels.
- MD are either successful or blocked (e.g. by an intrusion protection system).
- Demands are serialized (i.e. at most one demand is applied at a time)
System under-attack (2)

- Each successful malicious demand may introduce new “failure regions” on the demand space of the attacked channel.
- If more than 1 malicious demand succeeds, the union of the respective failure regions is added to the demand space of the affected channel.
System under-attack (3)

- If the failure regions introduced by attacks on both channels overlap (i.e. there are demands that belong to a failure region in both channels), then system $pfd$ increases by the size of the overlap.

- Malicious demands can be:
  - "Independently" applied to channels (i.e. the adversary is unaware that they are dealing with a 1-out-of-2 architecture);
  - Even with independent demands system $pfd$ can go up – a random overlap of failure regions caused by malicious demands applied to both channels;
  - "Synchronized", when the adversary is aware that they attack a two channel system
    - In the extreme case, the same set of new failure regions will be added to the demand space of both channels, possibly with some delay (we assume that simultaneous attacks on both channels are impossible).
System maintenance

- Channels can be *periodically* maintained.
- We consider two types of maintenance
  - “Cleansing”
    - similar to rejuvenation, but more than it – restoring the installation *from a clean copy*; rejuvenation is typically just a reboot.
    - Cleansing eliminates all failure regions introduced by successful malicious demands on the demand space of the particular channel.
  - Patching
    - Similar to cleansing, but also:
      - Reduces the initial channel $pfd$ (possibly the system $pfd$, too) – since *some bugs may have been fixed*
      - May also reduce the probability of success of some malicious demands (possibly to 0) by *eliminating exploitable vulnerabilities*.
System maintenance (2)

- System modelled as a stochastic state machine with an evolving “channel state” (the state is the set of failure regions added and removed by malicious demands and maintenance):
System maintenance (3)

- Maintenance with a 2-channel software must be managed to avoid maintaining more than one channel at a time:
  - mutex of channel maintenance
  - During maintenance, the system is reduced to a single channel
    - Hence the system \( pfd \) becomes equal to the \( pfd \) of the remaining operational channel

- Maintenance regimes \textit{can be combined} in 4 different scenarios:
  - No maintenance (this is the reality!)
  - Cleansing only
  - Patching only
  - Cleansing and patching
A probabilistic model was built to study the effects of attacks on the channels of a 1-out–of-2 system
- The SAN formalism was used with some custom C code added.
- Parameterization selected to allow for:
  - Different maintenance regimes
  - Different frequencies of maintenance
  - Different attack types (independent vs. synchronized)
- Looked at different measures of interest among them:
  - Mean Probability of failure over a time intervals
  - Mean Time to system failure (i.e. until both channels fail on the same demand)
    - In the studies I fixed the frequency of the normal demands.
- The model is solved via Monte Carlo simulation
  - 350 days of operation
  - Measures also calculated over 8 sub-intervals – to capture trends.
# Model parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR_size [day⁻¹]</td>
<td>Failure region sizes (uniformly distributed)</td>
<td>1.00E-04 – 2.00E-4</td>
</tr>
<tr>
<td>attackRateIncrease [day⁻¹]</td>
<td>Attack rate <em>increase</em> over time Changes to rates are applied in the model upon channel patching.</td>
<td>0.001</td>
</tr>
<tr>
<td>attackSProb_reduction</td>
<td>Coefficient of reduction of the malicious demand probability of success.</td>
<td>0.95</td>
</tr>
<tr>
<td>ch1_pfd</td>
<td>Channel 1 <em>pfd</em> (no malicious demands)</td>
<td>0.002</td>
</tr>
<tr>
<td>ch2_pfd</td>
<td>Channel 2 <em>pfd</em> (no malicious demands)</td>
<td>0.002</td>
</tr>
<tr>
<td>cleansing_interval [day]</td>
<td>Intervals between cleanings of channels (if cleansing is enabled in the particular study)</td>
<td>1</td>
</tr>
<tr>
<td>common_pfd</td>
<td>System <em>pfd</em> (before any channel is compromised).</td>
<td>1.00E-04</td>
</tr>
<tr>
<td>delay_sync_attack [day]</td>
<td>Delay between channel attacks in case channels are attacked synchronously.</td>
<td>0.05</td>
</tr>
<tr>
<td>demand_rates [day⁻¹]</td>
<td>Malicious demands rates (exponential distributions)</td>
<td>0.01-0.07</td>
</tr>
<tr>
<td>demand_types</td>
<td>Number of demands in channel 1 and channel 2 (could differ).</td>
<td>10</td>
</tr>
<tr>
<td>m_demand_max_FRs</td>
<td>Maximum failure regions per malicious demand</td>
<td>10</td>
</tr>
<tr>
<td>normal_demands_rate [day]</td>
<td>Interval between normal demands on 2-channel system.</td>
<td>1</td>
</tr>
<tr>
<td>patchingInterval [day]</td>
<td>Interval between patches (exponential distribution)</td>
<td>0.15</td>
</tr>
<tr>
<td>patch_ch_pfd_reduction</td>
<td>Channel <em>pfd</em> reduction coefficient after patching.</td>
<td>0.9</td>
</tr>
<tr>
<td>patch_CC_pfd_reduction</td>
<td>System <em>pfd</em> reduction coefficient after patching a channel.</td>
<td>0.95</td>
</tr>
<tr>
<td>upgrade_duration [day]</td>
<td>Maintenance duration (fixed interval)</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Results
Results (2)
Summary of the observations

- The effect of attacks on fault-tolerant software depends very significantly on the *model of attacks*
  - *not surprising, but some insight obtained with the model*
- Assessment assuming independent attacks (including those by red-teams/pen-testing) may give *dangerously optimistic conclusion* about how good the 2-channel system is.
  - The Target environment must be carefully considered and if synchronized attacks cannot be ruled out – apply synchronized attacks in the assessment.
Summary (2)

- Malicious demands may lead to significant variation of system reliability over time:
  - All estimates of the probability of successful mission are much worse than what they should have been with the average system $pfd$. Variation of system $pfd$ is clearly harmful.
  - Established techniques for reliability assessment based on assuming a constant $pdf$ throughout the entire mission (say a year, etc.) need to account for this variability.

- The model is available at: http://openaccess.city.ac.uk/16700/.
Implications for other replication systems

Any replication scheme is guaranteed to work correctly under a number of assumptions:

- 1-out-of-2: at least one of the channels must be correct
  - Taking down a channel for maintenance makes the 2-channel system a mere 1-channel system.

A number of intrusion-tolerant architectures are based on a Byzantine agreement protocol + cleansing in different flavors ("proactive recovery", "proactive obfuscation", etc.). These rely on a number of assumptions:

- Replication is guaranteed to work correctly only if the number of compromised replicas is smaller than a given threshold
- Can this assumption be enforced? How?
  - If not, can it be violated. How likely is such violation of the assumptions? The presented model allows for some exploratory analysis to be undertaken.
Future work

- The presented model is a “conceptual one”.
  - Can be used to get insight, but is probably useless for reliability prediction.
- Can the model be improved and made suitable for predictions? Maybe.
- More seriously (work in progress)
  - One needs to validate the inherent assumption built in the model construction that attacks decrease reliability. This needs to be done more thoroughly than has been done so far.
  - Data needed for validation may pose serious difficulties:
    - we have several stochastic processes, and some are not fully observable.
Questions

- Thank you!