Assessing and Validating Trust in Systems & Networks

**Trust:** How is it specified? What do we measure? How is it measured?

► Dependability & Security differ in invocation nature (deliberate or operational) of a vulnerability!

We **DO** know how to deal with systems that are:
- application specific
- domain specific
- technology specific
- well structured
- …& customized solutions!

Do we know how to deal with these issues in systems that are **NOT**:
- application specific
- domain specific
- technology specific
- network/fault specific

The Future is Pervasive (De-centralized) Computing: **Scale-less, mixed-mode, mobile, ...**

- Open evolvable/heterogeneity of elements + interfaces!
- Mobility & ad hoc connectivity
- Is Trust (& security) a modular, composable, and/or incremental attribute for this env?

**Challenge:** How do we specify, provide and validate the desired well-structured levels of “trust” in an increasingly “un-structured design & operational environment”?
Quantifying Trust in OS's

Neeraj Suri & Andréas Johansson

Department of Computer Science
Technische Universität Darmstadt, Germany

www.deeds.informatik.tu-darmstadt.de
**Trust & Assessing Trust**

**Trust**: Degree of reliance one is willing to put into ...

- ...ability to withstand perturbations (aka Survivability/Robustness)
  - Stresses + invocation of vulnerabilities (intentional or operational)

**Quantification of Trust:**

- Identification of vulnerabilities
- Assessment of impact of vulnerability
  - Would be marvelous to use analytical techniques (at design time)
    - OS size & complexity?
    - Dynamic nature of OS?
    - Services/code unknown?

- Can we utilize error impact analysis/profiling?
  - quantitative measures, reproducible, black/grey box specs ...
OS Errors

- Data level errors in drivers

Linux Kernel 2.4.20
Number of Lines of Source Code (totally 4,268,802)

- arch 16.02%
- drivers 57.67%
- include 12.48%
- ipc 7.28%
- init 0.08%
- kernel 0.33%
- mm 0.36%
- net 5.36%
- fs 0.03%
- lib 0.20%
- include 0.08%
- net 0.36%
- arch 0.20%
Profiling: Propagation Analysis of Data Errors

Module Error Permeability
To what degree does a module let errors "pass through"?

Module Error Exposure
To what degree is a module "exposed" to propagating errors?

Error Impact/ Criticality
What effects/damage can errors potentially cause?
To what degree does an error affect an O/P service?
Error Flow Profiling: Service Error Permeability

- Measure one driver’s influence on one OS service

\[
Pr(j, k) = Pr(error \text{ in } s_j \mid error \text{ in } d_k)
\]
OS Service Error Exposure

- **Given:** application uses service set
- **How are these services influenced by driver errors?**

\[ E^j = \sum_{k \in D} P^{j,k} \]

- **Use to compare services**
Driver Error Diffusion

• Which driver affects the system the most?

\[ D^k = \sum_{j \in S} P^{j,k} \]

• Use to compare drivers
Application Profile

• Which OS services does an application depend on?

• For each application:
  - Criticality (C_k)
  - Services used (s_i's)
  - Impact of service error (i_i's)
Application Service Exposure

- Given an OS, drivers and the OS profile
- Given an application \( \text{APP}_k \) and its profile

\[
A_k = C_k \cdot \sum_{\forall s_i} (E_i \cdot i^i)
\]

- Allows comparing the effects on each application
Case Study: Windows CE .Net

- Targeted drivers
  - Serial
  - Ethernet
- FI at interface
  - Data level errors
- Effects on OS services
  - 4 Test applications

Failure classes:
- **Class NF**: No visible failure
- **Class 1**: Error, no violation
- **Class 2**: Error, violation
- **Class 3**: Crash/Hang
Error Propagation

- Ethernet driver
  - 42 imported
  - 12 exported
- Most Class 1 (NF)
- 3 Class 3 (crashes)

<table>
<thead>
<tr>
<th>OS Service</th>
<th>#Tests</th>
<th>NF</th>
<th>Fail. Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeLibrary</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LoadLibraryW</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>NdisAllocateMemory</td>
<td>20</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>VirtualCopy</td>
<td>16</td>
<td>2</td>
<td>14 0 0</td>
</tr>
<tr>
<td>KernelIoControl</td>
<td>18</td>
<td>5</td>
<td>13 0 0</td>
</tr>
<tr>
<td>VirtualAlloc</td>
<td>18</td>
<td>7</td>
<td>11 0 0</td>
</tr>
<tr>
<td>memset</td>
<td>15</td>
<td>6</td>
<td>9 0 0</td>
</tr>
<tr>
<td>NdisMSetAttributesEx</td>
<td>16</td>
<td>10</td>
<td>6 0 0</td>
</tr>
<tr>
<td>NdisMSetAttributesEx</td>
<td>16</td>
<td>10</td>
<td>6 0 0</td>
</tr>
<tr>
<td>NdisMRegisterInterrupt</td>
<td>17</td>
<td>11</td>
<td>6 0 0</td>
</tr>
<tr>
<td>RegOpenKeyExW</td>
<td>17</td>
<td>12</td>
<td>5 0 0</td>
</tr>
<tr>
<td>NdisOpenConfiguration</td>
<td>3</td>
<td>0</td>
<td>3 0 0</td>
</tr>
<tr>
<td>memcpy</td>
<td>11</td>
<td>8</td>
<td>3 0 0</td>
</tr>
<tr>
<td>CreateMutexW</td>
<td>5</td>
<td>3</td>
<td>2 0 0</td>
</tr>
<tr>
<td>NKDbgPrintfW</td>
<td>3</td>
<td>2</td>
<td>1 0 0</td>
</tr>
<tr>
<td>GetProcAddressW</td>
<td>6</td>
<td>5</td>
<td>1 0 0</td>
</tr>
<tr>
<td><strong>Cumulative</strong></td>
<td>187</td>
<td>103</td>
<td>80 1 3</td>
</tr>
</tbody>
</table>
Error Exposure

- Serial driver
  - 50 imports
  - 10 exports
- Clustering of failures

<table>
<thead>
<tr>
<th>OS Service</th>
<th>Class NF</th>
<th>Class 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness 1</td>
<td>384</td>
<td>27</td>
</tr>
<tr>
<td>CreateFile</td>
<td>384</td>
<td>27</td>
</tr>
<tr>
<td>GetCommState</td>
<td>384</td>
<td>27</td>
</tr>
<tr>
<td>GetCommTimeouts</td>
<td>384</td>
<td>27</td>
</tr>
<tr>
<td>SetCommTimeouts</td>
<td>384</td>
<td>27</td>
</tr>
<tr>
<td>ReadFile</td>
<td>384</td>
<td>27</td>
</tr>
<tr>
<td>WriteFile</td>
<td>384</td>
<td>27</td>
</tr>
<tr>
<td>CloseHandle</td>
<td>411</td>
<td>0</td>
</tr>
<tr>
<td>Correctness 2</td>
<td>411</td>
<td>0</td>
</tr>
<tr>
<td>SetCommState</td>
<td>411</td>
<td>0</td>
</tr>
<tr>
<td>strlen</td>
<td>411</td>
<td>0</td>
</tr>
</tbody>
</table>
Diffusion

Findings:
- Ethernet driver more severe errors
- Some failures at boot-up → driver wrapper
- Clustering observed

<table>
<thead>
<tr>
<th></th>
<th>Ethernet</th>
<th>Serial</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Experiments</td>
<td>414</td>
<td>411</td>
</tr>
<tr>
<td>#Injections</td>
<td>228</td>
<td>187</td>
</tr>
<tr>
<td>#Class NF</td>
<td>330 (80%)</td>
<td>377 (92%)</td>
</tr>
<tr>
<td>#Class 1</td>
<td>80 (19%)</td>
<td>25 (7%)</td>
</tr>
<tr>
<td>#Class 2</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>#Class 3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$D_{C1}^k$</td>
<td>0.616</td>
<td>0.460</td>
</tr>
<tr>
<td>$D_{C2}^k$</td>
<td>0.002</td>
<td>0.022</td>
</tr>
<tr>
<td>$D_{C3}^k$</td>
<td>0.007</td>
<td>0</td>
</tr>
</tbody>
</table>
OS Trust Assessment

- **OS Error Flow Profiling**
  - Permeability
  - Exposure
  - Diffusion

- **Error Impact Analysis**

- **Limitations**
  - FI / Error Model
  - Application activation

- **Issues**
  - Error model granularity
    - Black → Gray → White
  - Target diversity
    - Applications, Drivers, OS's
  - Can robustness analysis & enhancements be conducted at run-time to meet desired level of (run-time) trust specifications?