How to detect untrusted code?

- System call monitoring
- Inline reference monitors

External reference monitor:-
- Monitor is outside application.
- High confidence of catching every code.
- Low performance as compared to inline reference monitor

Inline reference monitor:-
- Code should execute checks before accessing the resources
- May miss out few statements.
- High performance as compared to external reference monitor
- Involves complexity
- Less context switch
- Must do self detection
- Problem of buffer over flow, self modifying code etc. exists.

Example of checks:
- Checks should be done before accessing every resource.
Monitor_Check(open,fname,mode)
open(fname,mode)
Q. Who inserts the checks?
a. User inserts the checks

*In native client paper:*
Programmer -> code -> compiler -> binary -> verifier -> user
- Compiler inserted monitor checks.
- The verifier can be trusted by the user.
- Programmer driven checks

- In inline reference monitor, Programmer, code, compiler and binary are treated as untrusted. User applies checks to for his assurance.

**Monitor for memory read and writes:**

- We need to make sure that untrusted code writes ONLY to untrusted data
- Trusted code can make a call to a function in untrusted code
- We check the address where the function is writing data into.

```
cmp $r1, BASEADDR
blt abort
cmp $r1,TOPADDR
bgt abort
ld %r0,[%r1]
```

Diagram:

- Top Address (TOPADDR)
- Base Address (BADDR)
- Untrusted data
- Untrusted code
- Trusted data
- Trusted code
- Read/Write
- Read only
- Safety Check
- Safety forcing can be done in the way similar to baggy bounds checking.
- The untrusted data and code memory segment is of size $2^L$ and aligned.

```plaintext
AND %r1, ((1<L)-1)
OR  %r1,BASEADDR
Ld  %r0,[%r1]
```

Explanation of the above code:
X is the base address where r1 is pointing to and Y is the offset from that address.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
</tbody>
</table>

After AND

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Y</td>
</tr>
</tbody>
</table>

After OR

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BASEADDR</td>
<td>Y</td>
</tr>
</tbody>
</table>

We make sure that X should have the value of BASEADDR.

Q. There can be indirect jumps to the code which writes to trusted location. How to avoid that?

Answer:
- Divide the memory into chunks of some size (say 32 bytes)
- All jumps should be at the start of the segment
- Monitor operations should be inside that chunk
- No jumps are allowed inside the chunk
- Chunks are NOT allowed to span