Static Analysis For Security

<table>
<thead>
<tr>
<th>*soundness</th>
<th>CQual: Data</th>
<th>Medl/MECA/Xgti/Coverity</th>
</tr>
</thead>
<tbody>
<tr>
<td>=won’t miss a bug</td>
<td>Sound*</td>
<td>Control</td>
</tr>
<tr>
<td>No False Negatives</td>
<td>Unsound</td>
<td>Local</td>
</tr>
<tr>
<td>Whole program</td>
<td>Local</td>
<td>Lower FP</td>
</tr>
<tr>
<td>High False Positives</td>
<td>Many types of bugs</td>
<td></td>
</tr>
<tr>
<td>One type of bugs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CQual Static Analysis

| -require spec | +catch bugs early |
| - libraries | +no run time overhead |
| -language specific | +high coverage |
| -slow | +reduce testing time |
| -system specific | +low error |

Simplest Rule: Type Must Match

CQual: untainted tainted

```c
$ tainted int x;
$ untainted int y;
Y=x;  // error
X=y;  // OK
Int printf ($untainted char*);
Int printf($tainted char*);
$tainted char*s;
Printf(s);  //error
Tprintf(“Hello”);  //ok

$tainted int x;
$untainted int y;
```
$
\text{int } z; $

$Z=x; \text{// } \text{tainted } \leq \text{Qz}$

$Y=z; \text{// (tainted) } \leq \text{Qz } \leq \text{untainted} \text{ error}$

$Z=x; \text{// tainted } \leq \text{Qz } \leq \text{untainted} \text{ OK}$

Path = maybe a bug, but no path = no bug

Char * strcpy( char *dst, char *src) {

Char * tmp;

While (*src)

    *dst++=*src++;
    Return tmp;

}

$tainted$ char * s;

Char * t;

Strcpy (t, s);

Printf(t);

$untainted$ sanitize($tainted$ char * t){

... 

... // clean up t 

...
...return(*$untainted char *) t;
}

$untainted          $tainted

Web                 Data
Server

$untainted          $tainted

$untainted          $tainted

Data
Base