THE PROBLEM: Much (binary) software code possessed by end-users has been ***analyzed*** and ***tampered with***.

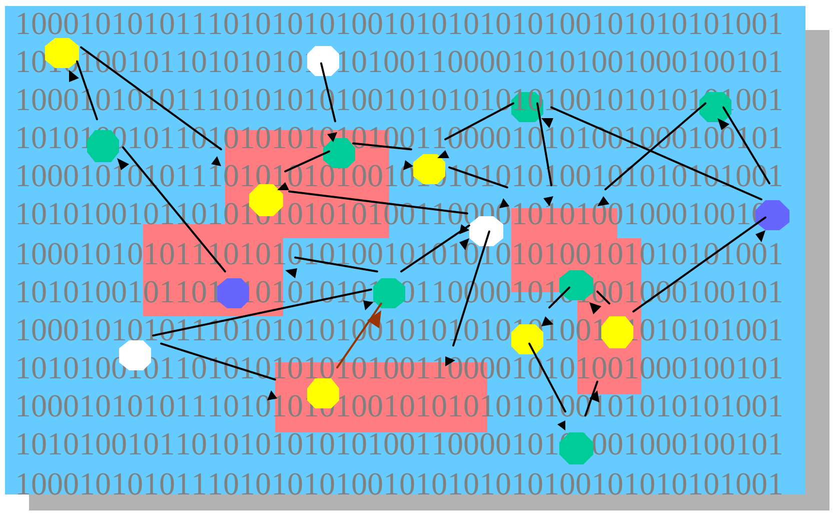
Example: Bypass software registration

```
 ...
  call registration
  if (status == OK)
     jmp main_module
  else
     exit
  ...
```

```
 ...
  nop; ...; nop
  if (OK)
     jmp main_module
  else
     exit
  ...
```

Program self-protection with various *guards*.

🟢 **Checksum** 🔵 **Repair** 🟡 **Others**



Possible consequences of tampering

- Program becomes unusable
- Program works as if not tampered with
- Error reporting
- etc.

If no tampering, guards are transparent to users of the software.

Why attacking guards NOT easy:

- No single points of attack
- Guards can execute only occasionally
- Guards can act stealthily
- Protection topology can vary across different copies of same software

Our software tamperproofing prototype

- Installs guards into Windows binary programs in an automated manner
- Able to deploys different guarding schemes
- Script-driven tamperproofing
- Currently works with VC++6.0

CERIAS

- Mike Atallah
- John Rice
- Tim Korb
- Hoi Chang